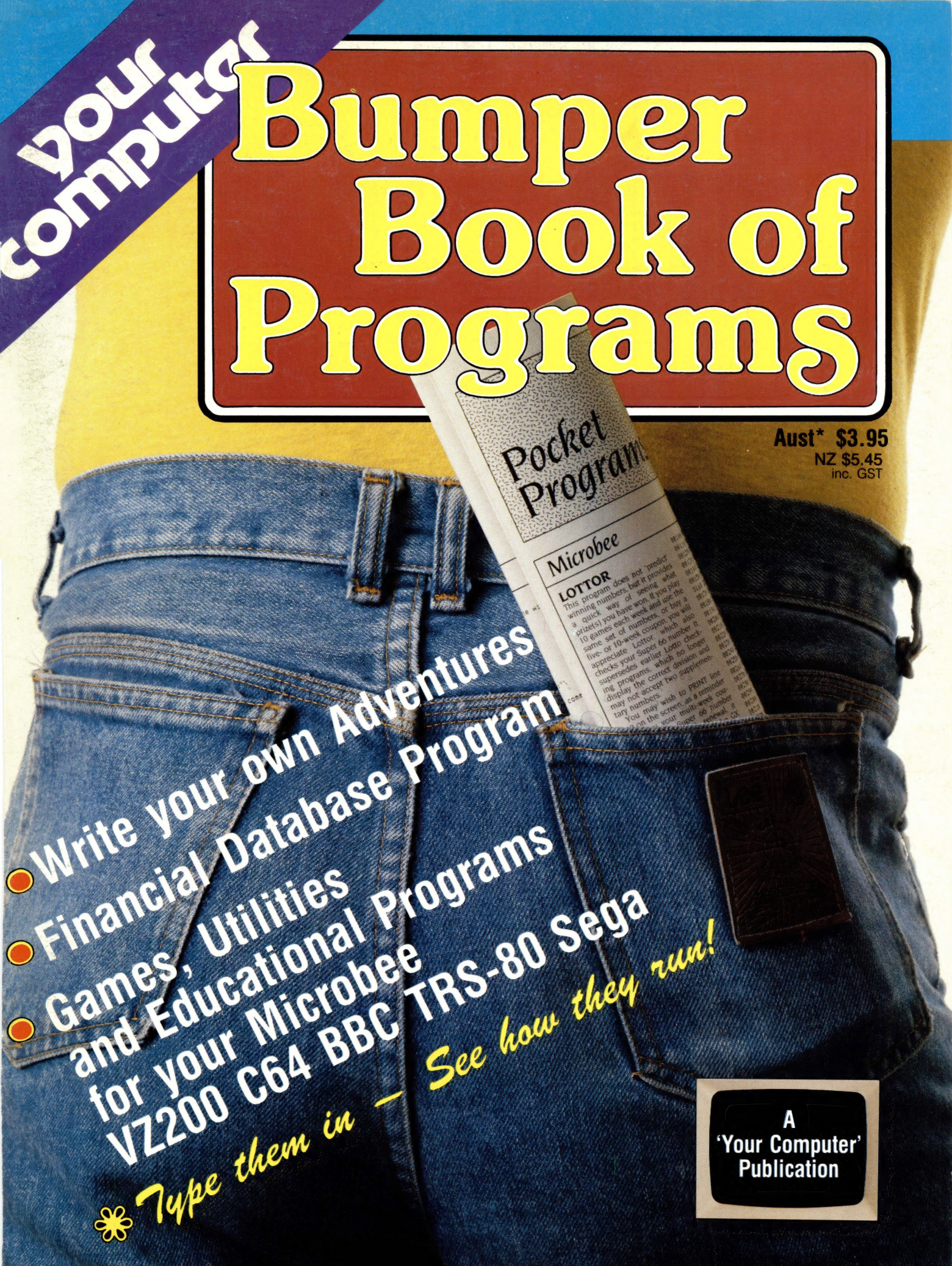
Your computer

Bumper Book of Programs



Aust* $3.95
NZ $5.45
inc. GST

- Write your own Adventures
- Financial Database Program
- Games, Utilities
- and Educational Programs
- for your Microbee
- VZ200 C64 BBC TRS-80 Sega

Type them in — See how they run!

A 'Your Computer' Publication

CONTENTS

EDITOR
Natalie Filatoff
CONSULTING EDITOR
Les Bell
MASTERMIND
Matt Whelan
PRODUCTION EDITOR
Jane Mackenzie
MANAGING EDITOR
Leo Simpson
PUBLISHER
Michael Hannan

OFFICE SERVICES
Carmel Trucillo
SUBSCRIPTION ENQUIRIES
Julie Plummer
ADVERTISING SALES
Damien Prins (Advertising Manager), and Craig Rowe (NSW Sales); David Farrington (Victorian Sales)
ADVERTISING PRODUCTION
John Crichton

OFFICES
NSW
180 Bourke Road, Alexandria 2015; (02) 693 6666.
Telex: FEDPUB AA74488
Victoria
150 Lonsdale St, Melbourne 3000; (03) 662 1222.
Telex: FEDPUB AA34340
Western Australia
Jim Wells, John Fairfax and Sons, 454 Murray Street,Perth 6000; (09) 481 3171.
Telex: AA94382 POSPE
Queensland
Warren Tapner, Federal Publishing, 26 Chermside St, Newstead 4006; (07) 854 1119.
Telex AA145520.
South Australia
Dane Hanson, John Fairfax and Sons 101 Waymouth St, Adelaide 5000; (08) 212 1212.

YOUR COMPUTER
is published monthly by the Federal Publishing Company Pty Ltd.
Printed by ESN — The Litho Centre Alexandria 2015; (02) 693 6666.
Editorial and NSW Advertising:
180 Bourke Road, Alexandria 2015.
Telex: FEDPUB AA74488.
Distributed nationally by Magazine Promotions.
*Recommended and maximum price only.


WAYNE!
TURN THAT BLOODY GAME OFF!
©B.J.AKHURST. 86.

CREATING YOUR OWN ADVENTURES

If you've ever wanted to write your own adventure programs, but have been discouraged by the amount of programming involved, take heart. In this article, Tim Hartnell gives you an empty 'adventure shell', which you can use as the framework for an infinite number of totally original adventures.

Why would intelligent people (such as you and I!) with access to massive amounts of computer power which could be doing something useful, want to bring their brainpower to bear on the problems of creating and exploring mythical landscapes? When your computer could be earning its keep telling you your cheque account is overdrawn, why are you forcing it to control a game in which you battle fierce monsters in labyrinthine caverns, invoke magic spells, and uncover vast hordes of elven gold?

The answer is obvious. Using your computer for serious things all the time is just plain boring. Trekking across alien landscapes, chopping up people with broadswords or axes and haggling with quasi-humanoid creatures in off-world taverns seems to many people a much better way to spend their time and their computer power.

But, if you decide you want to *write* your own adventure programs, instead of just playing commercial software adventures, you suddenly come up against one stubborn fact. If your adventure is going to be even half-way decent, an awful lot of meticulous programming is required to make the thing work.

That's where this article comes in.

The adventure shell is a sort of empty bucket into which you can pour your own imagination. The shell handles all the boring (but vital) bits, such as:

- Accepting the player's input and interpreting it.
- Acting on commands.
- Invoking monsters, getting them to fight you, deciding who has won a battle.
- Keeping track of whether or not the player is carrying a weapon or weapons, and the effect this can have on the outcome of a fight with a monster.
- Controlling movement around the adventure environment (whether it be inside the new Parliament House in Canberra, or in an underground tunnel below Ayer's Rock).
- Picking up and dropping treasure, and keeping track of what the player is carrying and how much it is worth.

The program will do all this, so you can save your energy to do such things as entering the PRINT statements which describe the adventure locations, the names of monsters, the kind of treasure and weapons hidden in various places within the adventure environment, and so on.

The shell is very flexible in its handling of player input. You can easily modify it to understand words which I have not catered for. As well, if you want a particular condition to be satisfied before the adventure can end (such as reaching a certain location, slaying a certain number of monsters, finding a certain number, or specified value, of treasured objects), it is extremely easy to include this.

Now, I'll not pretend you can just type the program in, play with it for an hour or so, and end up with a commercial-standard adventure. A fair degree of concentration is required to make sure you create a world which is geographically *consistent*, so the player can — even though it may take a great deal of effort — map your adventure world, and find that next time the world is visited, it still conforms to the map.

Map-making is one of the adventure player's skills and delights. To produce an adventure which is satisfying to play, your map must be consistent. The adventure shell looks after the map control, to ensure this consistency.

Using the Program

First of all, of course, you have to type the thing into your computer. In its present form, with lots and lots of REM statements, it occupies just under 13 Kbytes on my IBM PC, so it is likely to take around the same amount of space on your machine, whatever make it happens to be.

The REM statements are in this program for two reasons. The first is, naturally enough, to guide you through the program, so you know what each bit does. The second role is to 'make space' for you to add your own material. To see this, look at the lines from 860, and around 2700, in the listing. These allow you to get the shell to recognise additional words from the player input which have not been provided by the original program (although the shell, as it is, recognises all the important ones, controlling movement, fighting, and picking up and dropping objects).

Similarly, the REM statements from 3130 make space for you to put your descriptions of the adventure locations, or 'rooms', in which all the action takes place.

Therefore, I suggest you enter it just as it is, REMs and all, and save it in that form. Then all you have to do is load in the original shell whenever you wish to create a new adventure. Many of the REM statements are not referenced by GOTOs or GOSUBs, so they can be deleted from your ▶

final version of a particular adventure.

It might be worth getting your user club to make the shell available to members, to save everyone in the club having to type a copy in for their own use. The program can — either in its 'empty' form, or when changed into an adventure — be added freely to public domain collections. If you like, you can even sell programs produced using it. Although there will be no royalty payable on this, we would appreciate a mention that the shell was used, and that it came from *Your Computer* magazine, in the listing and/or documentation.

So, get the program into your computer, and then come back to this article to find out how to use it.

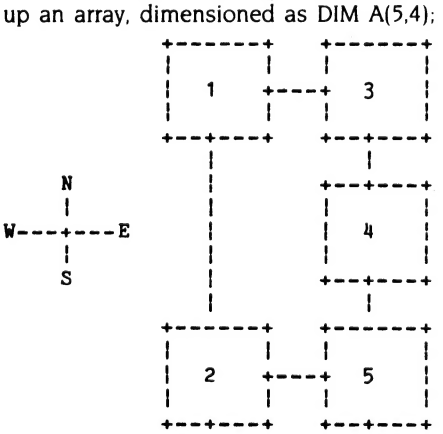
The Map

The first step in building an adventure program is to construct an environment which can be both mapped and represented in some way which the computer can store. You'll be pleased to know it is relatively easy to satisfy both these conditions.

Look at the five-room environment (map one). This is a very simple one, which we will treat as if it were an adventure environment.

The key to holding an environment like this in a way your computer can understand and manipulate is to set up an array, each element of which represents a room. The lines between rooms in diagram one are directions in which you can travel.

If you were in room one, you could move east into room three, or south into room two. In room four you can move north into room three and south into room five, and so on. Imagine we have set up an array, dimensioned as DIM A(5,4);



Map 1.

The adventure shell is a sort of empty bucket into which you can pour your own imagination. The shell handles all the boring (but vital) bits.

the first dimension is the room number, and the second is the four possible directions from that room, (that is, north, south, east and west).

Building a Travel Table

Armed with the map of the five-room environment, we can now build up what we in the adventure trade call a 'travel table', which can be fed into the array, to allow us to move from point to point within the environment.

Here's the travel table for the simple, five-room environment in map one:

Room	N	S	E	W
1	0	2	3	0
2	1	0	5	0
3	0	4	0	1
4	3	5	0	0
5	4	0	0	2

Take time to study this table, and the way it relates to the map, because it is probably the single most important key to building adventures using the shell.

Look at the table for room one. Under the 'N' (for north) column, you'll see a zero, which means you can't move north from room one (a fact which is easily verified by looking at our map). However, under the 'S' we see the number two, meaning if we travelled south from room one we would end up in room two (again, you can verify this from the map). Move east (the 'E' column, of course) from room one, and you'll end up in room three. The zero in the 'W' column means there is no travel possible west from room one.

You can work right through the table, if you like, checking the numbers on it correspond to the 'reality' of the map.

Now, to allow the player to move around the environment, we only need to (a) fill each element of the array with the relevant information from the travel table; (b) tell the player where he or she is; and (c) allow the decisions entered by the player regarding the direction in which he or she wants to move to be checked against the array. Then the player's new location has to be recorded, ready for the next move.

Using the shell, it is much easier to do this than you might think. In its present form, the shell caters for 16 rooms, and the DATA given (around line 4300 in the listing) is for a particular map which we will look at in due course. You can easily add more rooms, or use less if you want to do so.

We'll look at a simple program for controlling the five-room environment of map one, and once you see how this works, you'll be in a strong position to understand how to use the shell for a larger environment.

A simple program can be created to feed the relevant information from a travel table into an array:

```
10 DIM A(5,4)
20 FOR B=1 TO 5
30 FOR C=1 TO 4
40 READ A(B,C)
50 NEXT C
60 NEXT B
70 DATA 0,2,3,0
80 DATA 1,0,5,0
90 DATA 0,4,0,1
100 DATA 3,5,0,0
110 DATA 4,0,0,2
```

As you can see, the DATA statements correspond exactly to the items in our travel table.

If we decide the room (or cave, or location) which the player is currently occupying is to be designated by the variable RO (as we do in the shell), we could tell the player where he or she was as follows, as well as indicating which exits existed:

```
100 PRINT "YOU ARE IN ROOM
NUMBER";RO
110 IF A(RO,1)< >0 THEN PRINT
"A DOOR LEADS NORTH"
120 IF A(RO,2)< >0 THEN PRINT
"THERE IS AN EXIT TO THE SOUTH"
130 IF A(RO,3)< >0 THEN PRINT
"YOU CAN LEAVE VIA THE EAST EXIT"
140 IF A(RO,4)< >0 THEN PRINT
"A DOORWAY OPENS TO THE WEST"
```

The player's input, using the shell, is in the form of a two-word verb/noun phrase, such as 'Go north', 'Get diamond' or 'Slay monster'.

There is a 'string-parsing' routine, which strips the input down to two words, and sets the variable C$ equal to the second word. So the system knows if the first word is go, move or run, the second word must be the direction in which the player wishes to move. The shell only looks at the first three letters of the second word, as these are enough to discover the player's intentions.

Once the player has entered his or her decision, and the program has set C$ equal to the first three letters of the direction, the simple program we're working on here could check to see if an exit existed in that direction:

```
200 IF C$="NOR" AND A(RO,1)=0
THEN PRINT "YOU CANNOT MOVE
THAT WAY"
210 IF C$="SOU" AND A(RO,2)=0
THEN PRINT "YOU CAN'T WALK
THROUGH WALLS"
220 IF C$="EAS" AND A(RO,3)=0
THEN PRINT "TRY ANOTHER
DIRECTION"
230 IF C$="WES" AND A(RO,4)=0
THEN PRINT "THERE IS NO EXIT TO
THE WEST"
```

All that would be needed now would be a routine to go back for another input if the movement was not possible, and you'd have the bare framework of an adventure program. Of course, the shell does all this handling for you.

Once a valid input for direction has been accepted by the program, the movement itself takes place.

Note, by the way, that the room numbers are never referred to explicitly, as they are for the computer's internal consumption only. All the player reads is a description of the room: "You are in the dismal cellar, with old copies of *Your Computer* mouldering on the floor beside you . . . ". The description can include information on the exits ("You see a tunnel leading off to the north") or it can be left up to the player to find them by blundering about.

Back in our little five-room environment, imagine the player was in room four (look at map one) and entered the command 'Go north'.

The computer would proceed as follows, once it was sure the move was a legal one. Firstly, the variable RO would equal 4 (the room the player is currently occupying). The computer sees the move wanted is 'nor' and converts that, in its electronic head, to 1 ('sou' would become 2, 'eas' becomes 3, and 'wes' 4), so the computer knows the player is about to enter the room number A(RO,1). It would have been A(RO,2) if the player was moving south, and so on.

Although the rooms exist only on paper, and as elements in an array, the fact that they behave like 'real rooms' allows them to be perceived as though they were solid and real in a way which is uncanny.

As the room the player is currently in is room 4 (that is, RO equals 4), the computer simply looks at A(4,1) to see where the player is moving. In this case, it finds that A(4,1) equals 3, which is where a player would move by travelling north from room 4 (check it on the map).

The variable RO is then set equal to the new room, 3, and the process of moving through the environment continues.

Consistency and Reality

Although the rooms exist only on paper, and as elements in an array, the fact that they behave like 'real rooms' allows them to be perceived as though they were solid and real in a way which is uncanny.

Add descriptions of each room — "You are in a small workman's hut in the backyard of the Lodge. By peeping through the door of the hut you can see the Prime Minister walking purposefully towards you. You turn around, and see a trapdoor in the far corner" — and you'll find the environment takes on quite solid dimensions in your mind.

A Map for the Shell

If you look at the DATA statements near the end of the shell listing, you'll see a travel table I set up which you can use for your first adventure, if you like. Map two shows the map this table is based on. As you can see, it is not one which can be easily solved. You start in room 9, and the end of the adventure occurs when you reach room 16 (which need not, of course, be a 'room', but can instead be 'outside the castle', 'at the mouth of the cave', 'safe back in your undersea home again', or whatever you choose).

For your first practice with the shell, I suggest you use my map, giving the rooms your own names. Once you've given them names, you can use my treasures, and my monster names, and you'll instantly have a genuine adventure you can actually play.

It is extremely easy to test whether your travel table DATA is correct. Just get into the adventure environment, with your map, and then 'wander around it', to ensure it agrees with the map.

So you should copy map two, and give names to all the rooms. Perhaps a 'haunted castle' is as good an initial scenario as any other, and so the rooms can be called things like 'The Great Hall', 'The Castle Treasury', 'The Cruel Dungeon' and so on.

Once you've written in the names of the rooms on your map, you need to add some descriptive text for each room. Just suppose you had called room 12 'The Ma- ▶

NOTES:

gician's Quarters'.

Most of the line numbers from 3120 on are REM statements just waiting for your room descriptions to be inserted. So, we move through this set of numbers looking for the heading 'Room Twelve' (which is at line 3670). You can use any of the immediately following lines, up to the return, for the description (and, of course, you can add lines between these in this section if you need more space). Then, when you play the adventure and you are in room 12, the correct description will automatically be printed on the screen.

Your description could read:

"You find yourself in the Magician's Quarters, where years ago weird spells were concocted. Exits leave to the north, east and west."

So to this point in this article we've learned how to create a travel table, how to change it into DATA statements, and how to put room descriptions in the relevant positions within the shell.

You do not yet, however, have every bit of knowledge you need to use the program.

Attributes

As the player, you have six 'attributes' — strength, charisma, dexterity, intelligence, wisdom and constitution. If any of these falls to zero, the adventure is over, and you die.

These are gradually depleted as time goes on, to ensure you do not simply spend an endless amount of time wandering through your adventure environment. Your final score is related to a number of factors, including how strong your attributes are at the end of the game.

Each monster you meet and fight is also blessed with six attributes. When you meet a monster, the fight begins like this:

"Look out! There is an Embihuund here! What do you want to do?"

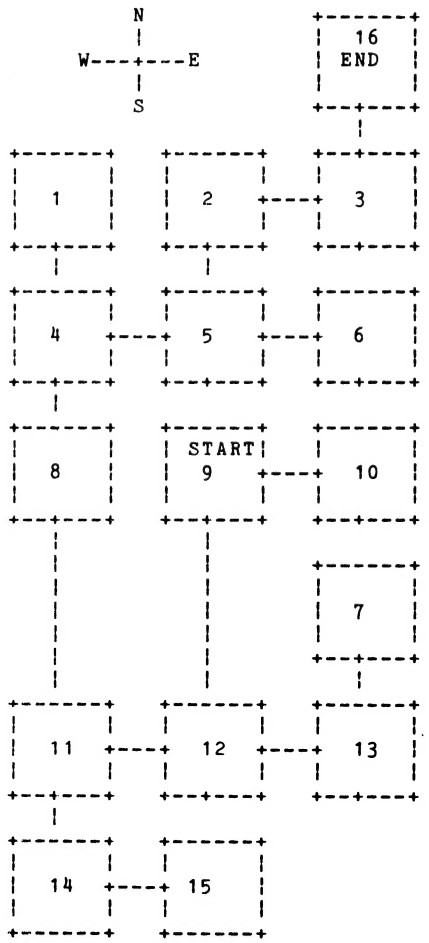
"Fight."

"Fight is just one word. I need two. What do you want to do?"

"Fight Embihuund."

"Your opponent is an Embihuund. The Embihuund's danger level is 6. You must fight the Embihuund with your bare hands. The Embihuund has the following attributes:

1 — Strength 6
2 — Charisma 15
3 — Dexterity 11
4 — Intelligence 13
5 — Wisdom 12
6 — Constitution 11.



Map 2.

Your attributes are:

1 — Strength 11
2 — Charisma 10
3 — Dexterity 10
4 — Intelligence 14
5 — Wisdom 16
6 — Constitution 10

Which attributes will you fight with – (2)?"

"1, 5"

As you can see, you are asked to enter your choice of the two attributes with which you will fight. Fairly obviously, you should choose the ones in which you most exceed the monster you are facing. The difference between your attributes and those of the monster, the 'ferocity factor' of the monster, and whether or not you are armed (with an axe, a sword, or both) all play a part in determining how well you will do in the coming battle.

Fortunately, the shell 'stage manages' the fight for you. As the fight progresses, your attributes are reduced if the monster scores a blow against you. At the end of the fight, the program reports the result, adding to your attributes if you have won.

Treasure and Terror

It is very simple to distribute the treasure throughout the adventure environment, as the shell does it for you.

If you look at the end of the listing, from line number 4180, you'll see that first the shell reads the names of the treasure and its value from DATA statements (and the two weapons, the sword and the axe, are given values of zero, so their worth won't be added to your growing fortune as you work your way around the environment). Then, with the routine from line 4210, the objects are distributed.

If you don't want a treasure in a particular room, all you have to do is add a line between 4220 and 4230 which reads as follows (where X is the number of the room which you want to remain without a treasure):

4225 IF Z=X THEN 4220

You'll see I've given you a set of sample treasures, which you can leave in for your first adventure, so you can concentrate on naming the rooms and getting to know how the shell works.

The monsters are distributed the same way, using the routine from line 4270. Again, if you want to ensure some of the rooms do not have monsters in them, a line to reject certain values, like the one used for treasure, can be inserted as line 4305.

The Vocabulary

It's fairly obvious which words the shell can act on at the moment. These are:

Kill or **Fight** (followed by the name of a monster).

Go, Move or **Run** (followed by a direction, such as east),

Get, Take or **Lift** (followed by the name of an object; you can carry up to five objects).

Drop, Put or **Leave** (followed by the name of an object; a room can hold up to three objects).

Help (doesn't do much good in the shell's present form, but you can add some real help if you like).

Quit (if you want to exit the adventure before getting to the end).

If you want to add additional words, you put these in lines 860 to 950, and then use the designated subroutines (2700, 2730, 2770, 2810 and 2850) to act on these additional words.

Your First Adventure

In summary, then, to create your first adventure, copy out map two and name the rooms. If you don't want to go to the trouble of writing full descriptions, you can just add lines like "You are in the drawing room", or even just the name of the room, such as "Mouldy cellar".

Leave my monster and treasure names in place. Wander around the map for a while, use the vocabulary as provided, and once you are familiar with the program in its present form, you can experiment by adding your own map, treasures and monsters.

Once you're happy with that, you can work on adding new material.

Note line 110, which sets QU to the value of one, and sends action to line 2950 if the player has reached room 16. This is the final room, and this line is the one which checks if the adventure has ended. Of course, once you start creating your own travel tables, you should change the 16 in this line into the number of the final room. (The starting room number, 9 in this case, is set equal to RO when the program begins, with line 4070.)

Further Ideas

Once you're familiar with the shell in the form it is presented here, there are several other ideas you might like to incorporate. These include:

- A monster which does not stay passively in one room, but follows you relentlessly once you've woken it up.
- A few other 'pseudo-players' under the computer's control which appear to be exploring the environment, too. The player can meet these people from time to time, possibly getting information about future dangers from them.
- A proper help command for the player to invoke if he or she feels hopelessly lost. A severe penalty (such as losing half the player's gold) should be charged to ensure this command is not used frequently.
- Doors can be locked, impassable, stuck or traps. Walls can fall in on players, floors give way to a gaping crevasse, and so on. Pictures can slide from walls, hitting our hapless player on the head. All these effects can be controlled within the subroutine which holds the room description, and a flag used to make sure the danger is not repeated if the room is revisited during one particular run of the adventure

I'd be extremely interested in seeing what you come up with using the shell. Please write to me care of *Your Computer*. May the Dreaded Ice-Dragon not molest you, and may all your chests be filled with Elven Gold.

```
10 REM      ADVENTURE SHELL
20 REM ************************
30 REM      By Tim Hartnell
40 REM  Interface Publications
50 REM ************************
60 GOSUB 3950:REM INITIALISE
70 REM ************************
80 REM REPORT TO PLAYER
90 FOR Z=1 TO 1000:NEXT Z
100 CLS
110 IF RO=16 THEN QU=1:GOTO 2950
120 REM ************************************
130 GOSUB 3060:REM ** ROOM DESCRIPTIONS **
140 REM ** NEXT LINE ACTIVATED IF OBJECT IN ROOM **
150 IF A(RO,5)<>0 OR A(RO,6)<>0 OR A(RO,7)<>0 THEN GOSUB 3850:REM OBJECTS
160 IF A(RO,8)=0 THEN 210:REM ** NO MONSTER IN ROOM **
170 PRINT TAB(3);"LOOK OUT!":PRINT "THERE IS AN ";M$(A(RO,8));" HERE!"
180 IF RND(1)>.7 THEN PRINT M$(A(RO,8));" ATTACKS!":KW=1:GOSUB 1670:GOTO 80
190 REM ************************************
200 REM ** NEXT LINES DECREMENT ATTRIBUTES **
210 IF RND(1)>.94 THEN ST=ST-1:IF ST<0 THEN ST=0
220 IF RND(1)>.94 THEN CH=CH-1:IF CH<0 THEN CH=0
230 IF RND(1)>.94 THEN DE=DE-1:IF DE<0 THEN DE=0
240 IF RND(1)>.94 THEN IT=IT-1:IF IT<0 THEN IT=0
250 IF RND(1)>.94 THEN WI=WI-1:IF WI<0 THEN WI=0
260 IF RND(1)>.94 THEN CO=CO-1:IF CO<0 THEN CO=0
270 REM ** NEXT LINES REPORT ATTRIBUTES TO PLAYER **
280 PRINT:PRINT "Your attributes are:"
290 PRINT TAB(4);"Strength -"ST"  Charisma -"CH
300 PRINT TAB(4);"Dexterity -"DE"  Intelligence -"IT
310 PRINT TAB(4);"Wisdom -"WI"  Constitution -"CO
320 REM ** NEXT LINE CHECKS IF ANY ATTRIBUTE IS ZERO **
330 IF ST*CH*DE*IT*WI*CO<>0 THEN 380
340 PRINT:PRINT "Unfortunately, you are exhausted..."
350 PRINT "so this adventure must end":QU=2:GOTO 2990
360 REM ************************************
370 REM ** NEXT LINES CHECK TO SEE IF PLAYER CARRYING ANYTHING **
380 FLAG=0
390 FOR J=1 TO 5
400 IF P(J)<>0 THEN FLAG=1
410 NEXT J
420 IF FLAG=0 THEN 490
430 CASH=0
440 PRINT:PRINT "You are carrying:"
450 FOR J=1 TO 5
460 IF P(J)<>0 THEN PRINT TAB(4);O$(P(J)):CASH=CASH+V(P(J))
470 NEXT J
480 IF CASH>0 THEN PRINT TAB(8);"Total value - $";STR$(CASH)
490 PRINT
500 REM ************************************
510 REM ** NEXT LINES ACCEPT PLAYER INPUT **
```

Microbee

CATTER

Catter is one of those annoyingly addictive games. The idea is to eat all the mushrooms, but Catter gets longer each time you indulge, and it tends to wander if not well controlled. Result? Very hard to survive the eight rounds — well, on level 3 it is.

Richard Larkin
Dee Why, NSW

```
00100REM CATTER 13/4/85 RICHARD LARKIN.

00110 CLS : NORMAL : PRINT "CATTER"\"YOU ARE IN CHARGE OF A CATTERPILLAR WHICH I
S RATHER FIDGETY."\"IT NEVER STOPS! SO YOU HAVE TO, AS BEST YOU CAN, HEAD IT TOW
ARDS";

00120 PRINT "ITS MEAL, MUSHROOMS, BUT ONLY THE RIPE ONES."\"USEING THE 'A','Z','
(' AND ')' KEYS CONTROL THE CATTERPILLARS"\"DIRECTION. IF YOU DON'T IT WILL!"

00130 PRINT "GAME ENDS WHEN THE CATTERPILLAR CANNOT MOVE."\"OR IF YOU SURVIVE 8
ROUNDS."\"GOOD LUCK!" : POKE 162,30 : POKE 163,128 : POKE 257,2

00140 FOR X=-2016 TO -2001 : READ Y : POKE X,Y : POKE X+16,Y : POKE X+32,Y : POK
E X+48,Y : NEXT X

00150 FOR X=-1952 TO -1937 : READ Y : POKE X,Y : POKE X+16,Y : POKE X+32,Y : POK
E X+48,Y : NEXT X

00160 FOR X=-1888 TO -1793 : READ Y : POKE X,Y : NEXT X : FOR X=-3072 TO -3009 :
 POKE X,255 : NEXT X

00170 DATA 3,12,16,40,83,64,137,138,138,137,64,83,40,16,12,3

00180 DATA 192,48,8,20,202,2,145,81,81,145,2,202,20,8,48,192

00190 DATA 0,0,15,48,64,64,128,128,159,98,2,4,4,8,8,15

00200 DATA 0,15,63,127,127,255,255,255,126,3,7,7,15,15,15,0

00210 DATA 1,3,7,15,31,63,127,255,255,127,63,31,15,7,3,1

00220 DATA 128,192,224,240,248,252,254,255,255,254,252,248,240,224,192,128

00230 DATA 0,0,240,12,2,2,1,1,249,70,64,32,32,16,16,240

00240 DATA 0,240,252,254,254,255,255,255,70,192,224,224,240,240,240,0

00250 DIM P(15),Z(3),D(3),N(3)

00260 FOR X=0 TO 3 : READ Z(X),D(X) : N(X)=X : NEXT X : PRINT "ANY KEY TO CONTIN
UE."

00270 DATA 130,-64,131,2,132,64,133,-2

00280 T=1 : R=1 : S=0 : I=USR(32774) : CURS 21,12 : PRINT "SKILL LEVEL (1 TO 3)"

00290 F=52-ASC(KEY) : IF F)3 OR F(1 THEN 290

00300 Q=24 : U=0 : Y0=0 : CLS : NORMAL : CURS 19,1 : PRINT "CATTER-by Richard La
rkin." : CURS 27,8 : PCG : PRINT "ROUND="R

00310 L=8 : FOR X=-3076-L*2 TO -3074 STEP 2 : POKE X,133 : POKE X+1,137 : NEXT X
 : E=-3074 : H=-3076-L*2 : G=3

00320 I0=FLT(R)/FLT(F)+1 : CURS 23,9 : PRINT " SCORE =          " : FOR X=0 TO 15

00330 P(X)=-4096+INT(RND*512)*2 : IF PEEK(P(X))=32 THEN POKE P(X),138 : POKE P(X
)+1,142 : NEXT X ELSE 330

00340 Q=Q-1 : IF Q=0 : POKE P(U),138+T : POKE P(U)+1,142+T : Q=3 : T=1-T

00350 CURS 31,9 : PRINT S" " : S=S-1 : C=G : A=PEEK(258) : IF A=26 : G=2 ELSE IF
 A=1 : G=0 ELSE IF A=44 : G=3 ELSE IF A=46 : G=1

00360 IF H+D(G))-3073 OR H+D(G)(-4096 : G=C

00370 O=H : H=H+D(G) : IF H()P(U) THEN 380 ELSE POKE H,32 : POKE H+1,32 : S=S+10
0 : U=U+1 : PLAY U : H=O : G=C : Y0=Y0+I0 : T=0 : Q=2 : IF U(16 THEN 340 ELSE 43
0

00380 IF PEEK(H)=32 AND H)-4096 AND H(-3073 THEN 400

00390 S=S-5 : H=O : FOR X=3 TO 0 STEP -1 : A=INT(RND*FLT(X+1)) : B=N(A) : N(A)=N
(X) : N(X)=B : IF PEEK(H+D(B))=32 OR H+D(B)=P(U) : G=B : NEXT*X 370 ELSE NEXT X
 : GOTO 450

00400 POKE O,Z(G) : POKE O+1,Z(G)+4 : POKE H,140 : POKE H+1,141

00410 IF Y0)1 AND Q=1 : Y0=Y0-1 ELSE LET C=PEEK(E) : POKE E,32 : POKE E+1,32 : E
=E+D(C-130)

00420 OUT (2),255 : FOR Y=0 TO 64-U*4 : NEXT Y : OUT(2),0 : GOTO 340

00430 R=R+1 : PLAY 9:6:8:4:7:2 : IF R(=8 THEN GOTO 300

00440 NORMAL : CURS 25,10 : PRINT " WELL DONE! " : CURS 19,11 : PRINT " YOU SURV
IVED 8 ROUNDS! " : PLAY 9:1:8:2:7:3:6:4:5 : GOTO 280

00450 NORMAL : PLAY 1,10 : CURS 27,10 : PRINT "BAD LUCK" : GOTO 280
```

NOTES:

Adventure Shell

```
2445 KK=K
2450 FOR Z=1 TO 700:NEXT Z
2460 PRINT "THE ";G$;" -"MT
2470 PRINT "      YOU -"HT
2480 PRINT "----------------------------------------"
2490 IF K=0 THEN PRINT "You struck a splendid blow!":MT=MT-1
2500 IF K=1 THEN PRINT "THE ";G$;" STRIKES OUT":HT=HT-1:CH=CH-1
2510 IF K=2 THEN PRINT "YOU DRAW THE ";G$;"'S BLOOD":ST=ST-1
2520 IF K=3 THEN PRINT "You are wounded!!":IT=IT-1
2530 IF K=4 THEN PRINT "THE ";G$;" IS TIRING":DE=DE-1
2540 IF K=5 THEN PRINT "You are bleeding...":WI=WI-1
2550 IF K=6 THEN PRINT "YOU WOUND THE ";G$:MT=MT-1
2560 IF RND(1)>.25 AND HT>0 AND MT>0 THEN FOR T=1 TO 1600:NEXT T:GOTO 2440
2570 IF HT<MT THEN 2600
2580 PRINT "YOU HAVE SLAIN THE ";G$
2590 ST=ST+2:DE=DE+2:WI=WI+2:CH=CH+2:IT=IT+2:CO=CO+2:MK=MK+2:GOTO 2670
2600 PRINT "THE ";G$;" GOT THE BETTER OF":PRINT "YOU THAT TIME..."
2610 IF Z=1 OR Q=1 THEN ST=4*INT(ST/5)
2620 IF Z=2 OR Q=2 THEN CH=3*INT(CH/4)
2630 IF Z=3 OR Q=3 THEN DE=6*INT(DE/7)
2640 IF Z=4 OR Q=4 THEN IT=2*INT(IT/3)
2650 IF Z=5 OR Q=5 THEN WI=5*INT(WI/6)
2660 IF Z=6 OR Q=6 THEN CO=INT(CO/2)
2670 A(RO,8)=0
2680 FOR Z=1 TO 500:NEXT Z
2690 RETURN
2700 REM OWN COMMANDS ACTED ON HERE
2710 REM OWN COMMANDS ACTED ON HERE
2720 REM OWN COMMANDS ACTED ON HERE
2730 REM OWN COMMANDS ACTED ON HERE
2740 REM OWN COMMANDS ACTED ON HERE
2750 REM OWN COMMANDS ACTED ON HERE
2760 REM OWN COMMANDS ACTED ON HERE
2770 REM OWN COMMANDS ACTED ON HERE
2780 REM OWN COMMANDS ACTED ON HERE
2790 REM OWN COMMANDS ACTED ON HERE
2800 REM OWN COMMANDS ACTED ON HERE
2810 REM OWN COMMANDS ACTED ON HERE
2820 REM OWN COMMANDS ACTED ON HERE
2830 REM OWN COMMANDS ACTED ON HERE
2840 REM OWN COMMANDS ACTED ON HERE
2850 REM OWN COMMANDS ACTED ON HERE
2860 REM OWN COMMANDS ACTED ON HERE
2870 REM OWN COMMANDS ACTED ON HERE
2880 REM ***********
2890 REM END OF GAME
2900 PRINT
2910 SC=0:REM SCORE
2920 IF QU<>4 THEN 2950
2930 PRINT "I did not imagine you would turn"
2940 PRINT TAB(5);"out to be a quitter!":GOTO 2990
2950 PRINT:PRINT "CONGRATULATIONS! You have completed"
2960 PRINT TAB(7);"THE ADVENTURE"
2970 SC=100
2980 PRINT:PRINT
2990 SC=99*(SC+20*CASH+47*MK+ST+2*CH+3*DE+4+IT+5*WI+6*CO)/QU
3000 IF MK>0 THEN PRINT TAB(7);"YOU KILLED"MK"MONSTERS"
3010 PRINT:PRINT TAB(7);"YOU FOUND $";STR$(CASH);" WORTH"
3020 PRINT TAB(8);"OF TREASURE":PRINT
3030 PRINT:PRINT "Your score for this Adventure is"SC
3040 END
3050 REM *****************
3060 REM ROOM DESCRIPTIONS
3065 PRINT "YOU ARE IN ";
3070 IF RO<9 THEN ON RO GOSUB 3090,3140,3190,3240,3290,3340,3390,3440
3075 IF RO>8 THEN ON RO-8 GOSUB 3490,3540,3590,3640,3690,3740,3790
3080 RETURN
3090 REM ** ROOM ONE **
3100 PRINT "ROOM ONE"
3110 REM
3120 REM
3130 RETURN
3140 REM ** ROOM TWO **
3150 PRINT "ROOM TWO"
3160 REM
3170 REM
3180 RETURN
3190 REM ** ROOM THREE **
3200 PRINT "ROOM THREE"
3210 REM
3220 REM
3230 RETURN
3240 REM ** ROOM FOUR **
3250 PRINT "ROOM FOUR"
3260 REM
3270 REM
3280 RETURN
3290 REM ** ROOM FIVE **
3300 PRINT "ROOM FIVE"
3310 REM
3320 REM
3330 RETURN
3340 REM ** ROOM SIX **
3350 PRINT "ROOM SIX"
3360 REM
3370 REM
3380 RETURN
3390 REM ** ROOM SEVEN **
3400 PRINT "ROOM SEVEN"
3410 REM
3420 REM
3430 RETURN
3440 REM ** ROOM EIGHT **
3450 PRINT "ROOM EIGHT"
3460 REM
3470 REM
3480 RETURN
3490 REM ** ROOM NINE **
3500 PRINT "ROOM NINE"
3510 REM
3520 REM
3530 RETURN
3540 REM ** ROOM TEN **
3550 PRINT "ROOM TEN"
3560 REM
3570 REM
3580 RETURN
3590 REM ** ROOM ELEVEN **
3600 PRINT "ROOM ELEVEN"
3610 REM
3620 REM
3630 RETURN
3640 REM ** ROOM TWELVE **
3650 PRINT "ROOM TWELVE"
3660 REM
3670 REM
3680 RETURN
3690 REM ** ROOM THIRTEEN **
3700 PRINT "ROOM THIRTEEN"
3710 REM
3720 REM
3730 RETURN
3740 REM ** ROOM FOURTEEN **
3750 PRINT "ROOM FOURTEEN"
3760 REM
3770 REM
3780 RETURN
3790 REM ** ROOM FIFTEEN **
3800 PRINT "ROOM FIFTEEN"
3810 REM
3820 REM
3830 RETURN
3840 REM *****************
3850 REM DESCRIBE OBJECTS
3860 PRINT
3870 PRINT TAB(3)"YOU CAN SEE:"
3880 IF A(RO,5)<>0 THEN PRINT TAB(4);O$(A(RO,5))
3890 IF A(RO,6)<>0 THEN PRINT TAB(4);O$(A(RO,6))
3900 IF A(RO,7)<>0 THEN PRINT TAB(4);O$(A(RO,7))
3910 FOR Z=1 TO 500:NEXT Z
3920 PRINT
3930 RETURN
3940 REM **********
3950 REM INITIALISE
3960 DIM A(16,8),P(5),O$(8),V(20),M$(8),T(5)
3970 REM ** REM NEXT LINES DECIDE
     HUMAN ATTRIBUTES **
3980 ST=INT(RND(1)*6+RND(1)*6+RND(1)*6)+3
3990 CH=INT(RND(1)*6+RND(1)*6+RND(1)*6)+3
4000 DE=INT(RND(1)*6+RND(1)*6+RND(1)*6)+3
4010 IT=INT(RND(1)*6+RND(1)*6+RND(1)*6)+3
4020 WI=INT(RND(1)*6+RND(1)*6+RND(1)*6)+3
4030 CO=INT(RND(1)*6+RND(1)*6+RND(1)*6)+3
4040 CASH=0:REM TREASURE
4050 RO=9:REM STARTING ROOM
4060 QU=1:REM END OF GAME FLAG
4070 MK=0:REM MONSTERS KILLED
4080 C$=""
4090 REM ** SET UP ROOMS **
4100 FOR X=1 TO 16
4110 FOR Y=1 TO 4
4120 READ A(X,Y)
4130 NEXT Y
4140 NEXT X
4150 REM ** DISTRIBUTE TREASURE **
4160 FOR Z=1 TO 8
4170 READ O$(Z), V(Z)
4180 NEXT Z
4190 FOR Q=5 TO 8
4200 Z=INT(RND(1)*15+1)
4210 IF A(Z,5)<>0 THEN 4200
4220 A(Z,5)=Q:REM OBJECT NO. IN ROOM
4230 NEXT Q
4240 PRINT
4250 REM ** DISTRIBUTE MONSTERS **
4260 FOR J=1 TO 8
4270 READ M$(J)
4280 Z=INT(RND(1)*15+1)
4290 IF A(Z,8)<>0 THEN 4280
4300 A(Z,8)=J
4310 NEXT J
4330 CLS
4340 RETURN
4350 REM ****************************
4360 REM ROOM DATA
4370 DATA 0,4,0,0:REM ROOM ONE
4380 DATA 0,5,3,0:REM ROOM TWO
4390 DATA 16,0,0,2:REM ROOM THREE
4400 DATA 1,8,5,0:REM ROOM FOUR
4410 DATA 2,0,6,4:REM ROOM FIVE
4420 DATA 0,0,0,5:REM ROOM SIX
4430 DATA 0,13,0,0:REM ROOM SEVEN
4440 DATA 4,11,0,0:REM ROOM EIGHT
4450 DATA 0,12,10,0:REM ROOM NINE
4460 DATA 0,0,0,9:REM ROOM TEN
4470 DATA 8,14,12,0:REM ROOM ELEVEN
4480 DATA 9,0,13,11:REM ROOM TWELVE
4490 DATA 7,0,0,12:REM ROOM THIRTEEN
4500 DATA 11,0,15,0:REM FOOM FOURTEEN
4510 DATA 0,0,0,14:REM ROOM FIFTEEN
4520 DATA 0,3,0,0:REM ROOM SIXTEEN
4530 REM ** OBJECT DATA **
4540 DATA "RING",567
4550 DATA "KEY",2
4560 DATA "LOCKET",15
4570 DATA "ELVEN-GOLD",799
4580 DATA "SWORD",0,"AXE",0
4590 DATA "AMYTHEST",27
4600 DATA "CRYSTAL",45
4610 REM ** MONSTER NAMES **
4620 DATA "ENTANGLER"
4630 DATA "ARTIFACTUM"
4640 DATA "INKBLOTT"
4650 DATA "UGLY UNDEAD"
4660 DATA "'ORRIBLE ORC"
4670 DATA "ICE-DRAGON"
4680 DATA "EMBIHUUND"
4690 DATA "INSALIVATE"
```

LEDGERMASTER

BY LINDSAY R. FORD
Dreamcards Software

This commercial package, written by Lindsay Ford of Dreamcards Software for the Microbee, will come in very handy around tax time — but it'll take a bit of typing . . .

WITH TAXATION time almost here again, many readers will be faced with the daunting prospect of having to shuffle through bundles of old receipts and cheque butts to put together their tax returns. This problem is a particular burden to people like me, who run small businesses that don't make enough to justify paying the fees of a good tax accountant. Club treasurers facing the task of preparing their annual reports will also know just what I mean!

One solution is to invest in a software accounts package, but these are generally disk-based, expensive to buy and more sophisticated than most of us need. LedgerMaster was written to overcome these problems. It is a financial database program designed to run on 32 Kbyte (or larger) Microbee computers and has all the features you're likely to need to prepare a profit and loss statement for taxation purposes or for publication in your club magazine. It allows you to create a ledger containing up to 320 transaction records ('files'), which can be stored on tape, printed, sorted and searched as with any general database.

Where LedgerMaster stands out, however, is it also conducts a balance of the ledger files (and of any other ledgers you have 'chained' to it) and compiles these into a proper profit and loss statement. This takes almost all the effort out of preparing tax returns or treasurers' reports, whether you're typing in the receipts and payments in one frenzied, last-minute burst or simply tallying a ledger you've created through the year as transactions occurred.

Program Entry

The program is really a compromise between efficient code and legibility. Ideally, a BASIC program needs as many instructions as possible crammed into each line, if maximum speed and economic memory use are to be achieved, but this would have made LedgerMaster extremely difficult to read. I've opted for slightly less-than-optimum line lengths, but even so the size of the program means you have to be extremely careful as you type it in. Please bear the following points in mind:

- Use AUTO mode to enter the program, as this minimises the risk of 'non-printing characters' being accidentally included in the code. If you have my BASIC Screen Editor fitted to your Bee, use it. This will avoid the problem entirely.
- REMs that appear in a line of code (for example in line 1) should be left out entirely.
- Where a line contains a REM but no BASIC code (for example lines 19 and 20), the REM should be entered, but any text following it omitted.
- The up-arrow character (↑) appears frequently in the program. Don't leave it out.
- Watch out for variables I, O and Q. These are easy to confuse with ones and zeros.

Once the program has been entered exactly as shown in the listing (subject to the above comments), use the GX command to locate all lines containing REMs. These lines should be deleted (using the DELETE command) and several copies of the program should then be stored on tape.

At this stage you have an almost complete program. Now type in the following (in 'direct mode' – don't give it a line number):

```
CLS: X=0: Y=PEEK(2258) + 256*PEEK(2259): FOR Z=2304 TO Y:
X=X+PEEK(Z): NEXT Z: PRINT "X="; X, "Y="; Y: END
```

When you press the <RETURN> key the screen will clear, and after a few minutes two numbers will appear. 'Y' is the address in decimal of the top of the BASIC file and should be 11819. If it's higher than this you've put something in the program that shouldn't be there, if lower you've left something out. 'X' is a checksum — it is the total of the numbers contained in each of the memory bytes making up the program. If it is -16146 then all is well and you can proceed to the next stage; otherwise you'll have to look for your typing error.

Stage 2 of the operation involves removal of unnecessary spaces in the listing, as these greatly reduce execution speed. Enter "GX/ /"<Return> and rest a brick on the full-stop key, then go and make yourself a cup of coffee. After about five minutes all spaces will have been removed. Now you'll have to insert spaces in the places they're meant to be by substituting a space for each up-arrow. Use "GX/↑/ /"to do this. Once done you should save a few copies on tape, then enter the test routine shown above. This time Y should be 10079 and X, 17639 If they are you've got a fully operational LedgerMaster.

Using the Program

When you RUN LedgerMaster there will be a brief delay as the main arrays are initialised, then a menu will appear giving you the option to 'Create Base' or 'Load'. You will also see two numbers at the bottom right of the screen. The first of these is the free memory counter, which tells you how much room remains in the ledger (you can't run out of file space on a 32 Kbyte machine), and the second is the number of free files available. As the ledger hasn't been initialised this second number will be zero. You won't have a ledger tape you can load at this stage, so let's look at the first option.

Create Base

This routine initialises the ledger, clearing out any existing material in memory. For this reason you will be asked if you are 'Sure?' if you access it when a ledger is already present. You will then be requested to name the ledger (maximum 24 characters), then to input the various categories of payments and receipts you want included on the final balance sheet (such as 'insurance', 'rent', 'royalties', 'wages' and so on). Give these a good deal of thought before starting, as they comprise the framework around which your financial report or tax return will be built.

You can enter up to 20 classes of payments and 20 of receipts (maximum 12 characters), and a 'carried forward' balance for each. This balance will be of importance if you are including amounts accrued from a previous period, such as debts outstanding from the last tax year. No 'carried forward' balance will be entered if you press the <Return> key. Pressing <Return> when you're prompted to input a payment or receipt category indicates you've finished entering items in that class.

Once the payments and receipts categories are entered your ledger 'base' is complete and the program will return to the menu. You will now have four options available — 'Create', 'Load', 'Enter' and 'Save'. If you

want to go on and input a couple of files, the first option to look at is 'Enter'.

Enter

Pressing key 3 clears the menu from the screen and the prompt at bottom left asks you to input the 'day' part of the date of the particular file. By now you'll have noticed the program uses a non-standard input routine, the length of the particular field being indicated by a row of star (*) characters next to the prompt. As each character is typed it replaces one of the stars, the entry being complete either when the <RETURN> key is pressed or when the last character in that field is typed.

With date entries the data field is two characters wide, so a date comprising one digit (for example the '1' in 1/12/84) can be entered either as '1 <RETURN>' or as '01'. Once the day is entered you will be prompted for the month and then the year. To avoid unnecessary typing, the program allows you to 'borrow' date data from the last file you typed. Thus, if the date on the last record was '14/3/84' and you press the <RETURN> key in answer to the 'day' prompt in the next record, it will be assigned the date '14/3/84' as well. This can also be used with parts of dates, so if you input '28' for the day, then press <RETURN> when prompted to input the month, the date will be entered as '28/3/84'.

An error check routine is incorporated to detect illegal dates (including February 29 on non-leap years). Note, as each field is entered the partially complete file is displayed at the top of the display window.

After the date is entered you are asked to enter 'From/To'. This identifies from whom the money was received or to whom it was paid (maximum 16 characters). Pressing the <RETURN> key will enter the word 'Paid' in this field. The next item requested is the receipt or invoice number (maximum seven characters), which can be omitted by pressing the <RETURN> key, and the word 'B/card' inserted by pressing ↑ B (that is, the <CONTROL> and 'B' keys simultaneously) or the word 'Cash' by pressing ↑ C.

The next two steps deal with the category into which the item will be inserted in the balance sheet. The first asks you if it is a receipt or payment (press 'R' or 'P'), then all categories included in the ledger base will appear in the display window, with letters to identify them. Press the key corresponding to the particular category into which the payment falls. If you press a key that doesn't correspond to one of the displayed categories it will be ignored.

The final item to enter is the amount of money involved. Although this field is nine characters wide, two spaces are reserved for cents and one for the decimal place. (If you're dealing with amounts of a million dollars or more you can afford to get an accountant to do it for you!) There's no need to input trailing zeros in the cents part, so '100 dollars' can be entered as '100 <RETURN>' and '87 dollars 30 cents' as '87.3 <RETURN>' Payments and receipts appear in separate right-justified columns in the display window: payments to the right and receipts to the left.

This completes your file entry and you will now be asked whether you wish to proceed to enter the next file (press <RETURN>), to return to the menu (key 'M') or to 're-do' (correct) the record you just typed (key 'R'). The program will not allow you to exceed the ledger capacity, so once the 320th file is typed it will return to the menu and announce 'Ledger Full'.

Save

This option allows you to save all or part of your ledger on tape. Pressing key 4 will take you into the 'SAVE' routine and you'll then be asked whether you want to save the base (key 1), the files (key 2) or the whole ledger (key 3). Pressing any other key aborts the routine, returning you to the menu.

In the majority of cases you will want to save the whole ledger on tape (key 3), but the base and file save facilities can be useful if you want to save only the files and incorporate a new base in the ledger with the 'Load' routine 'Merge' facility, or if you want to use the 'New Base' facility to 'Chain' a number of ledgers together (see below). Whatever option you choose, the program recognises the importance of your ledger files and saves two copies — just in case a fault in the tape causes a loading problem. The message 'Start Tape' will appear on the screen, followed by a short delay (to allow the tape leader to go past the record head), then the computer will announce it is saving 'Copy 1', 'Copy 2', then return to the menu.

The ledger 'header' (the important ledger information) is saved at 300 baud, then the base and files are saved at 1200 baud. This is not a job for bargain-basement tapes. I use TDK 120 us bias cassettes — anything less is just begging for an unloadable file that you'll have to type all over again. Database 'Save' routines tend to bring out the worst in Microbee cassette interfaces; readers whose Bees are 32 Kbyte IC models or earlier should consider making the tape I/O modifications suggested in the *Microbee Hacker's Handbook* (available through *Your Computer*), and those with the Telcom v1.0 or 1.1 communications ROM ought to either update to a later version or have the permanent RTC link disconnected. Otherwise, loading at 1200 baud becomes a game of chance!

Load

The load facility is designed for maximum flexibility in manipulating your ledgers. Not only can you load a full ledger (key 3) or a base or files alone (keys 1 or 2), but if there is already a ledger in memory and you choose to load only the files or base of your tape ledger, that section will replace the corresponding section of the ledger already in memory. This allows you to create a new base if the old one was deficient (such as if you decide an extra category ought to have been included in the payments section).

A 'Merge' facility (key 4) is also included, allowing you to add the files in your tape ledger to the end of the ledger (if any) already in memory. If this exceeds the ledger capacity (320 files), the computer announces "Merge - Part only", and you'll have to use the 'Display' function to check how many of the tape files were loaded before the ledger reached its capacity.

Once you've selected the load option you want to use, the program requests you to 'Start Tape', and this message remains on the screen until a valid tape header is encountered. At this stage the file title will be displayed with some additional information (such as whether the tape consists of a base or files only or whether the files are unsorted — see below). If the tape file is incompatible with your load command (such as if you commanded the computer to load a base and the tape dump is files only), the routine will abort leaving any existing ledger intact. Please note, limits on program length meant that no load indicator or error checking could be included, so if your ledger hasn't loaded after about two minutes then press <RESET> and try again.

Sort

This routine arranges the files in your ledger in chronological order according to the file dates. As LedgerMaster is written in BASIC the Sort routine is quite slow (about five minutes to sort 320 files), so you will be asked whether you are 'Sure?'. Press key 'N' to abort or key 'Y' to continue.

Your files don't have to be sorted before they're saved on tape or printed, but your final printout will be much easier to follow if a sort is first carried out. A counter will appear on the screen as the routine is in progress (just to reassure you your computer hasn't died of a stroke!).

Delete

This function is useful if you want to get rid of an incorrectly typed file or a slab of files in a ledger that has been partly 'merged' with an earlier ledger. Enter the number of the first file to be deleted and the file will appear in the display window. Now press key 'N' if you've picked the wrong one or key 'Y' if it's correct. You'll then be asked for the number of the last file to be deleted. If only one file is to be deleted, press the <RETURN> key, otherwise enter the appropriate file number and check that the file that ▶

appears on the display window is the last one in the section you want to erase. Both files that appeared on the screen will be deleted, along with all files in between. Invalid file numbers will be rejected.

Print

This is a powerful function with a range of options. You'll first be asked whether you want to search for any particular string, allowing you to search through the date, item, receipt number and/or money columns of the ledger for all occurrences of a particular set of characters. So if you entered 'B/card <RETURN>' in response to the 'Search' prompt, the only ledgers printed would be those containing this expression.

If you're searching the money column for all entries of a particular amount, please remember trailing zeros in the cents column are omitted. A hash mark (#) followed by a file number entered in response to the search request will cause printing to commence at that file number. To print all files in the ledger simply press the <RETURN> key.

The program will now ask whether you want the printout directed to the screen or to a printer (keys 'S' or 'P'). If you select the printer option a sub-menu will appear, asking you to select the printer type (press an invalid key if you chose the printer option by mistake). Once you've made your selection you'll be asked to press a key to continue. Check your paper is properly aligned in the printer, press a key and away it will go.

The ledger printout consists of three sections (see Figure 1). First are the categories 'carried forward' (if any), followed by the ledgers themselves and finally the balance sheet. Base categories with a nil balance are omitted. If the 'search' mode is used then only the ledgers are printed, as otherwise the balance sheet totals would be misleading.

When printing on the screen the ledger categories are shown simply by their code letter (to accommodate the 64-character-width screen), but these are printed in full if a printer is used. The printer format also includes a page heading showing the ledger name, column headings, page number and (if the ledger has been sorted) the dates it covers.

New Base

This option assists in 'chaining' ledgers by taking the base categories of an existing ledger, adding to each the total of all applicable files, then erasing the files themselves. This new base now contains 'carried forward' totals that are used in the new ledger, so its balance sheet reflects the total of the entries in both.

When the option is selected you'll first be asked if you're 'Sure?', as accidental selection of this routine would have disastrous results. You'll then be asked to enter the name of the new ledger, following which the necessary operations will be carried out and the program will return to the menu.

A Final Balance

No doubt some readers will have the utter horrors at typing in a program of this length. If you want to save yourself the trouble, forward a cheque or money order for $15, payable to Dreamcards, 8 Highland Court, North Eltham 3095 (mail order only), and I'll send you a tape.

Anyone wanting a more sophisticated tape-based accounting system might like to consider my Multibank program ($34.95 plus $1.50 postage and packing, for cassette and manual). This generates cassette files containing up to eight ledgers, and has heaps of frills usually only found on expensive disk-based systems (such as automatic allowance for periodical debits). □

LedgerMaster

```
00001 POKE 140,1  REM Disable <BREAK> Key
00002 GOSUB 333  REM Initialise Storage arrays
00003 IN#0: OUT#0  REM VDU On
00004 N=8: IF BO$(0)="" THEN LET N=2 ELSE IF C=1 THEN LET N=4
00005 KO$=KEY$: K4$="": I=0: Q=0: IF C<3 AND N=8 THEN LET N=5
00006 GOSUB 298: INVERSE: CURS 28,4: PRINT "^MENU^": NORMAL
00007 PRINT \ SPC(14) "Create^.....^1";
00008 PRINT SPC(4) "Load^.....^2": IF N=2 THEN 16
00009 PRINT SPC(14) "Enter^.....^3";
00010 PRINT SPC(4) "Save^.....^4": IF N=4 THEN 16
00011 PRINT SPC(14) "Delete^.....^5";: IF N=5 THEN 16
00012 PRINT SPC(4) "Sort^.....^6"
00013 PRINT SPC(14) "Print^.....^7";
00014 PRINT SPC(4) "New^Base^...^8"
00015 IF C=V+1 THEN PRINT \ SPC(21) "***^LEDGER^FULL^***"
00016 K2$="Select^Menu^Option": GOSUB 226
00017 X=INT(VAL(KO$)): IF X=0 OR X>N THEN 16
00018 PLAY 22,1: ON X GOTO 65,33,83,120,138,22,150,216
00019 REM
00020 REM ------------------ Sort Routine ------------------
00021 REM
00022 GOSUB 232: IF K1$="N" OR S=1 THEN 3
00023 X=0: GOSUB 317  REM Set-up Date array
00024 FOR X=1 TO C-1: GOSUB 292: A1(X)=F4: NEXT X
00025 FOR X=1 TO C-2: GOSUB 317: FOR M=X+1 TO C-1  REM Sort
00026 IF A1(M)=>A1(X) THEN 29
00027 KO$=AO$(M): AO$(M)=AO$(X): AO$(X)=KO$
00028 F1=A1(M): A1(M)=A1(X): A1(X)=F1
00029 NEXT M: NEXT X: S=1: GOTO 79
00030 REM
00031 REM ----------------  "Load" Routine ----------------
00032 REM
00033 N=3: IF C>1 AND C<V+1 THEN LET N=4
00034 GOSUB 324: IF O=0 THEN 3
00035 OUT#0 OFF: IN#2  REM VDU Off, 300 baud Cass. in
00036 INPUT K1$, D, H, X, Y, Z, E: K1$=K1$(;7)
00037 IN#0: OUT#0  REM VDU On again
00038 IF O=3 AND H<3 THEN LET O=0
00039 IF O=1 AND H=2 THEN LET O=0
00040 IF O>1 AND H=1 THEN LET O=0
00041 KO$="": J=1: L=D-1: IF O<4 THEN 44
00042 J=C: L=L+C: C=L: S=0: KO$="^-^Merge"
00043 IF C>V+1 THEN LET C=V+1: KO$=KO$+"^(Part^only)"
00044 IF O=2 OR O=3 THEN LET C=D: S=Z
00045 IF O=1 OR O=3 THEN LET BO$(0)=K1$: P=X: R=Y: T=E
00046 K1$=K1$+KO$: CURS 0,12: PRINT [A64^32]
00047 CURS (64-LEN(K1$))/2,12: PRINT K1$: CURS 26,13
00048 IF H=1 OR O=1 OR C=1 THEN PRINT "^(Base)": GOTO 51
00049 IF (H=2 OR O=2) AND C>1 THEN PRINT "^(File)": GOTO 51
00050 IF S=0 AND C>1 AND O>1 THEN PRINT "(Unsorted)"
00051 IF O=0 THEN 79 ELSE LET N=X: IF Y>X THEN LET N=Y
00052 CURS 0: OUT#0 OFF: IN#3  REM VDU Off, 1200 baud Cass. in
00053 FOR X=1 TO N  REM Load Base
00054 IF (O=2 OR O=4) AND H=3 THEN INPUT KO$, KO$, F1, F1
00055 IF O=1 OR O=3 THEN INPUT BO$(X), B1$(X), B2(X), B3(X)
00056 NEXT X: IF O=1 THEN 58 ELSE FOR X=J TO C STEP 3
00057 INPUT AO$(X), AO$(X+1), AO$(X+2): NEXT X
00058 IF C<V+1 AND O>1 THEN GOSUB 336
00059 Y=P+1: IF P<20 THEN GOSUB 337
00060 Y=R+1: IF R<20 THEN GOSUB 338
00061 GOTO 79
00062 REM
00063 REM --------- ------ "Create" Routine ----------------
00064 REM
00065 GOSUB 331: IF K1$="N" THEN 3
00066 G=24: K2$="File^Title": GOSUB 298: GOSUB 250: BO$(0)=K1$: I=1
00067 GOSUB 304: P=P+1: GOSUB 249: BO$(P)=K1$: K3$=K1$
00068 IF K3$="" AND P=1 THEN LET BO$(1)="Payment": GOTO 70
00069 IF K3$="" THEN LET P=P-1: GOTO 71
00070 GOSUB 237: B2(P)=F3: IF P<20 AND K3$<>"" THEN 67
00071 PLAY 22,1
00072 GOSUB 307: R=R+1: GOSUB 249: B1$(R)=K1$: K3$=K1$
00073 IF K3$="" AND R=1 THEN LET B1$(1)="Receipt": GOTO 75
00074 IF K3$="" THEN LET R=R-1: GOTO 76
00075 GOSUB 237: B3(R)=F3: IF R<20 AND K3$<>"" THEN 72
00076 PLAY 22,1: T=0: FOR X=1 TO 20: IF B2(X)>0 THEN LET T=T+1
00077 IF B3(X)>0 THEN LET T=T+1
00078 NEXT X: C=1: K3$="": GOTO 3
00079 CURS 0: PLAY 22,1; 0,5: GOTO 3
00080 REM
00081 REM ---------------- "Enter" Routine ----------------
00082 REM
00083 IF C>V THEN 3
00084 S=0: AO$(C)="": M=C: GOSUB 268: I=C-1: G=2
00085 IF C>1 THEN LET X=C-1: GOSUB 292
```

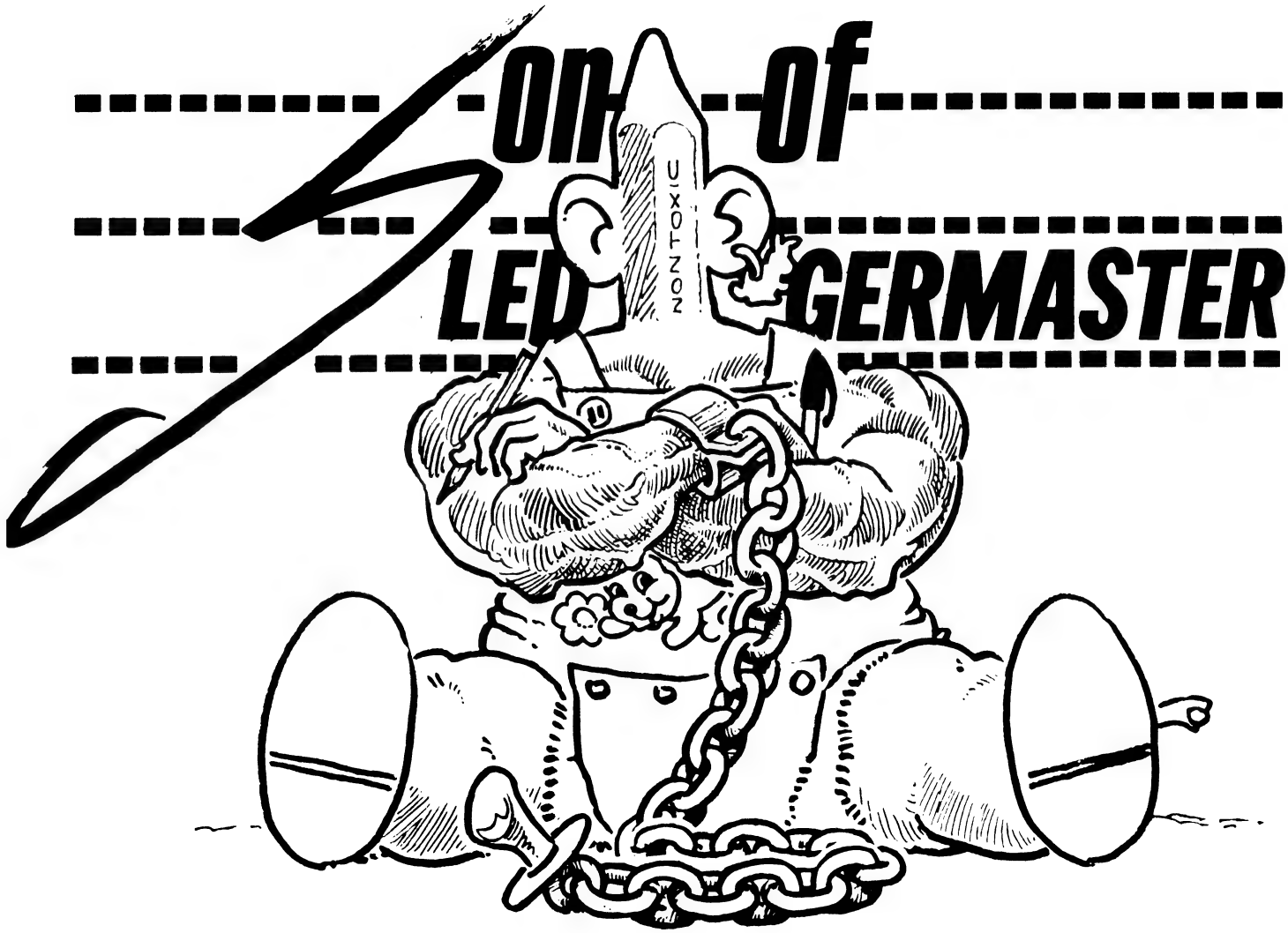
LedgerMaster

```
00086 K2$="Day": GOSUB 250: IF K1$="" AND F1>0 THEN 92
00087 F1=VAL(K1$): IF F1=0 OR F1>31 THEN 86
00088 K2$="Month": GOSUB 250: IF K1$="" AND F2>0 THEN 92
00089 F2=VAL(K1$): IF F2=0 OR F2>12 THEN 88
00090 K2$="Year": GOSUB 250: IF K1$="" AND F3>0 THEN 92
00091 F3=VAL(K1$): IF F3=0 THEN 90
00092 RESTORE: FOR X=1 TO INT(F2): READ Z: NEXT X
00093 DATA^31,28,31,30,31,30,31,31,30,31,30,31
00094 IF FRACT(F3/4)=0 AND F2=2 THEN LET Z=29
00095 IF INT(F1)<=Z THEN 97
00096 CURS 1,16: PRINT "***^DATE^ERROR^***";: PLAY 10,12: GOTO 86
00097 K0$=STR$(INT(F1)): K1$=K0$(;2)+"/"
00098 K0$=STR$(INT(F2)): K1$=K1$+K0$(;2)+"/"
00099 K0$=STR$(INT(F3)): IF F3<10 THEN LET K1$=K1$+"0"
00100 K1$=K1$+K0$(;2): GOSUB 267: K2$="From/To": G=16: I=1
00101 GOSUB 250: IF K1$="" THEN LET K1$="Paid"
00102 GOSUB 267: K2$="Cheque/Receipt^No.": G=7: GOSUB 250
00103 IF X=2 THEN LET K1$="B/card" ELSE IF X=3 THEN LET K1$="Cash"
00104 GOSUB 267: K2$="Receipt^(R)^or^Payment^(P)"
00105 GOSUB 226: IF K1$<>"R" AND K1$<>"P" THEN 105
00106 GOSUB 285: GOSUB 298
00107 IF K1$<>"P" THEN 109
00108 M=P: IF M=1 THEN LET K1$="A": GOTO 111 ELSE GOSUB 304: GOTO 110
00109 M=R: IF M=1 THEN LET K1$="A": GOTO 111 ELSE GOSUB 307
00110 K2$="Category": GOSUB 226: X=X-64: IF X<1 OR X>M THEN 110
00111 GOSUB 267
00112 G=9: I=0: K2$="Amount": GOSUB 250: GOSUB 242: IF X=1 THEN 112
00113 GOSUB 267: K2$="Next=<CR>^^Menu=M^^Re-do=R^":I=1
00114 GOSUB 226: IF K1$="R" THEN 84
00115 IF K1$<>"M" AND X<>128 THEN 114
00116 C=C+1: IF X=128 THEN 83 ELSE 3
00117 REM
00118 REM ----------------- "Save" Routine -----------------
00119 REM
00120 N=3: IF C=1 THEN LET N=1
00121 GOSUB 324: IF H=0 THEN 3
00122 N=P: IF R>P THEN LET N=R
00123 FOR Q=1 TO 2: PLAY 0,60: K0$=B0$(0)
00124 CURS 0,12: PRINT [A28^32]; "Copy"; Q; [A28^32]: CURS 0
00125 OUT#2  REM 300 Baud Cassette out (VDU remains on)
00126 PRINT "******"; K0$; ","; C; ","; H; ","; P; ","; R; ","; S; ","; T
00127 PLAY 0,1: OUT#3  REM Delay / 1200 Baud Cassette out
00128 IF H=2 THEN 130 ELSE FOR X=1 TO N  REM Save BASE
00129 PRINT B0$(X); ","; B1$(X); ","; B2(X); ","; B3(X): NEXT X
00130 PLAY 0,1: IF C<2 OR H=1 THEN 133
00131 FOR X=1 TO C STEP 3  REM Save FILES
00132 PRINT A0$(X); ","; A0$(X+1); ","; A0$(X+2): NEXT X
00133 OUT#0  REM De-select Cassette output
00134 NEXT Q: GOTO 79
00135 REM
00136 REM ---------------- "Delete" Routine ----------------
00137 REM
00138 G=3: K3$="^file^to^be^deleted": K2$="First"+K3$
00139 GOSUB 250: O=INT(VAL(K1$)): IF O<1 OR O=>C THEN 138
00140 M=O: GOSUB 268: GOSUB 232: IF K1$="N" THEN 3
00141 G=3: K2$="Last"+K3$+"^-^or^<CR>": I=1: GOSUB 250
00142 L=INT(VAL(K1$)): IF K1$="" THEN LET L=O
00143 I=0: IF L<O OR L=>C THEN 141 ELSE IF L=O THEN 145
00144 M=L: GOSUB 268: GOSUB 232: IF K1$="N" THEN 141
00145 K3$="": Z=L-O+1: X=0: GOSUB 317: FOR X=L+1 TO C
00146 A0$(O)=A0$(X): O=O+1: NEXT X: C=C-Z: GOSUB 336: GOTO 79
00147 REM
00148 REM ---------------- "Print" Routine -----------------
00149 REM
00150 G=20: I=1: H=1: K2$="String^search": GOSUB 250: K4$=K1$
00151 IF ASC(K4$)=35 THEN LET H=INT(VAL(K4$(;2))): K4$="#"
00152 IF H=0 OR H>C-2 THEN 150
00153 K2$="Screen^(S)^or^Printer^(P)": GOSUB 226
00154 L=12: IF K1$="S" THEN 163 ELSE IF K1$<>"P" THEN 3
00155 GOSUB 298: K2$="PRINTER^Type:": PRINT \\ SPC(15); K2$
00156 PRINT \ SPC(20) "Parallel^^^^^^-^^Key^1"
00157 PRINT SPC(20) "Serial^300bd^^-^^Key^2"
00158 PRINT SPC(20) "Serial^1200bd^-^^Key^3"
00159 PRINT SPC(20) "Abort^-^^any^other^Key"
00160 GOSUB 226: X=X-48: IF X<0 OR X>4 THEN 3
00161 L=57: Q=X: IF X>1 THEN LET Q=X+2
00162 GOSUB 225: OUT#0: OUT#0 OFF: OUT#Q ON: PRINT
00163 FOR X=1 TO 20: B4(X)=B2(X): B5(X)=B3(X): NEXT X
00164 D=0: F3=0: F4=1: O=T: U=0: W=0: E=C+P+R+T+4
00165 IF K4$<>"" THEN LET E=C: O=1
00166 G=1: K0$=KEY$: IF K0$="" THEN 169 ELSE PLAY 22,2
00167 IF K0$="A" OR K0$="a" THEN 210  REM Detect <A> key
00168 K1$=KEY$: IF K1$="" THEN 168  REM Hold if any other key
00169 IF D<>0 THEN 178 ELSE IF Q>0 THEN 171  REM Page Heading
00170 FOR X=3 TO 14: CURS 1,X: PRINT [A64^32]: NEXT X: CURS 1,3: GOTO 178
00171 D=2: PRINT "LEDGER:^"; B0$(0); SPC(26-LEN(B0$(0)));
00172 K2$="^to^": IF S=0 THEN 175  REM Print dates if SORTED
00173 K0$=A0$(1): GOSUB 286: K2$=K1$+K2$
00174 K0$=A0$(C-1): GOSUB 286: K2$=K2$+K1$: PRINT K2$;
00175 PRINT SPC(34-LEN(K2$)); "Page^"; F4: F4=F4+1
00176 PRINT "No.^^^^^DATE^^^^TRANSACTION"; SPC(17);
00177 PRINT "CLASS"; SPC(16); "AMOUNT^($)": PRINT [A79^45]
00178 IF O=0 OR K4$<>"" THEN 188  REM Print "Carried Forward"
00179 U=U+1: IF U>P THEN 182
00180 F1=B2(U): IF F1=0 THEN 179
00181 K0$=B0$(U): X=LEN(K0$): Z=0: GOTO 184
00182 W=W+1: F1=B3(W): IF F1=0 THEN 182
00183 K0$=B1$(W): X=LEN(K0$): Z=11
00184 IF O=T THEN PRINT "^CARRIED^FORWARD:^^^"; ELSE PRINT SPC(20);
00185 IF Q>0 THEN PRINT SPC(22); K0$; SPC(27-X-Z);
00186 IF Q=0 THEN PRINT K0$; SPC(33-X-Z);
00187 O=O-1: PRINT [F10.2^F1]: GOTO 205  REM Print amount carried
00188 IF K4$="" OR K4$="#" OR O<>1 THEN 190
00189 PRINT \ SPC(10); "SEARCH;^"; K4$ \: O=0: G=3: GOTO 205
00190 M=H: IF K4$="" THEN LET M=H-T
00191 IF M<1 OR M>=C THEN 194 ELSE GOSUB 269  REM Print FILE
00192 IF M=0 THEN LET B5(N)=B5(N)+F1 ELSE LET B4(N)=B4(N)+F1
00193 GOTO 205  REM Add balance in file to BASE category
00194 IF K4$<>"" THEN 205
00195 M=M-C: IF M=0 THEN PRINT: GOTO 205  REM FILE End
00196 IF M>P THEN 199 ELSE IF B4(M)=0 THEN 209  REM Payments
00197 PRINT SPC(20); B0$(M); SPC(22-LEN(B0$(M))); [F10.2^B4(M)]
00198 F3=F3-B4(M): GOTO 205
00199 M=M-P: IF M>R THEN 202 ELSE IF B5(M)=0 THEN 209  REM Receipts
00200 PRINT SPC(20); B1$(M); SPC(22-LEN(B1$(M))); [F10.2^B5(M)]
00201 F3=F3+B5(M): GOTO 205
00202 X=67: IF Q=0 THEN LET X=51  REM Print Total at end
00203 IF H=E OR H=E-2 THEN PRINT SPC(X+2); [A10^45]: G=1
00204 IF H=E-1 THEN PRINT SPC(X); [F11.2^F3]: G=1
00205 D=D+G: IF D<L THEN 209  REM Lines per page counter
00206 REM Delete next line if not using fan-fold paper
00207 IF Q>0 THEN PRINT \\\\\\\  REM Perf. gap = 7 lines
00208 D=0: IF Q>0 THEN 209 ELSE GOSUB 225: IF K0$="A" THEN 212
00209 H=H+1: IF H<=E THEN 166  REM Detect end of printing
00210 IF Q>0 THEN OUT#Q OFF: GOTO 212
00211 IF K0$<>"A" AND K0$<>"a" THEN GOSUB 225
00212 Q=0: GOTO 3
00213 REM
00214 REM ---------------- "New Base" Routine ----------------
00215 REM
00216 GOSUB 232: IF K1$="N" THEN 3
00217 I=1: G=24: K2$="New^Title": GOSUB 250: IF K1$<>"" THEN LET B0$(0)=K1$
00218 FOR Y=1 TO C-1: K0$=A0$(Y): X=SEARCH(K0$,"|",3): GOSUB 287
00219 O=ASC(K0$): GOSUB 286: Z=ASC(K0$)-64: GOSUB 286: F1=VAL(K0$)
00220 IF O=80 THEN LET B2(Z)=B2(Z)+F1 ELSE LET B3(Z)=B3(Z)+F1
00221 X=Y: GOSUB 317: NEXT Y: C=1: GOSUB 336: GOTO 76
00222 REM
00223 REM ------------- "Key to Continue" Sub. -------------
00224 REM
00225 K2$="KEY^to^Continue"
00226 G=1: GOSUB 250: X=ASC(K1$)
00227 IF X>96 AND X<123 THEN LET X=X-32: K1$=CHR$(X)
00228 RETURN
00229 REM
00230 REM ---------------- "Sure (Y/N)" Sub. ----------------
00231 REM
00232 K2$="Sure^(Y/N)": GOSUB 226: IF K1$<>"N" AND K1$<>"Y" THEN 232
00233 RETURN
00234 REM
00235 REM ---------- "Previous Balance" Input Sub. ---------
00236 REM
00237 G=9: K2$="Balance^Forward": GOSUB 250: GOSUB 242
00238 IF X=1 THEN 237 ELSE RETURN  REM Loop if invalid amount
00239 REM
00240 REM ---------------- "Money Input" Sub. ----------------
00241 REM
00242 X=1: F1=VAL(K1$): IF F1<=0 AND I=0 OR F1=>1000000 THEN 245
00243 X=0: F2=FRACT(F1): F3=F1-F2+FLT(INT(F2*100))/100
00244 K1$=STR$(F3): K1$=K1$(;2)  REM Round off to 2 places
00245 RETURN
00246 REM
00247 REM ----------------- Main "Input" Sub. ----------------
00248 REM
00249 G=12: K2$="Category"
00250 GOSUB 299: Y=LEN(K2$): CURS 1,15: PRINT [A64^45] [A63^32];
00251 X=V+1-C: IF X>V THEN LET X=V  REM Limit free files counter
00252 CURS 1,16: PRINT K2$;: CURS 52,16: PRINT INT(FRE($)); "^;"; X;
00253 CURS Y+3,16: Z=0: K1$="": FOR X=1 TO G: PRINT "*";: NEXT X
00254 CURS Y+1,16: PRINT "?^";  REM Print prompt line
00255 K0$=KEY$: X=ASC(K0$): IF X=124 OR X=128 THEN 255
```

```
00256 IF X=8 OR X=127 THEN 261 ELSE IF X>13 THEN 259
00257 IF Z>0 THEN CURS 0: RETURN  REM End if <CR>
00258 IF I>0 THEN LET K1$="": RETURN ELSE 255
00259 Z=Z+1: CURS Y+Z+2,16: PRINT K0$;  REM Print Input so far
00260 K1$=K1$+K0$: IF Z=G THEN RETURN ELSE 255  REM End of string?
00261 K0$=K1$(;1, LEN(K1$)-1): K1$=K0$  REM <DEL>ete Key
00262 IF Z>0 THEN CURS Y+Z+2,16: PRINT "*";: CURS Y+2,16: Z=Z-1
00263 GOTO 255
00264 REM
00265 REM ------------- "File entry" Print Sub. ------------
00266 REM
00267 M=C: GOSUB 285  REM "M" is record number
00268 GOSUB 298: PRINT "FILE:"
00269 IF K4$="" OR K4$="#" THEN 271  REM Implement "Search"
00270 K0$=A0$(M): IF SEARCH(K0$,K4$)=0 THEN LET G=0: RETURN
00271 K0$=STR$(M)+".": K0$=K0$(;2): X=LEN(K0$)
00272 PRINT K0$; SPC(4-X);: K0$=A0$(M): GOSUB 286  REM Date
00273 PRINT SPC(9-LEN(K1$)); K1$; "^";: IF Q>0 THEN PRINT "^";
00274 GOSUB 286: PRINT K1$; SPC(18-X);  REM Item
00275 GOSUB 286: PRINT K1$; SPC(9-X);  REM Cheque/Receipt No.
00276 GOSUB 286: M=0: IF K1$="P" THEN LET M=11  REM Rec/Pay?
00277 IF Q=0 THEN PRINT K1$; "^";  REM Don't print in full on VDU
00278 GOSUB 286: N=ASC(K1$)-64: K2$=B0$(N): IF M=0 THEN LET K2$=B1$(N)
00279 IF Q>0 THEN LET X=LEN(K2$): PRINT K2$; SPC(16-X); ELSE PRINT K1$;
00280 GOSUB 286: F1=VAL(K1$): IF F1>0 THEN PRINT SPC(M); [F10.2^F1]
00281 RETURN
00282 REM
00283 REM ------------ Extract file section Sub. -----------
00284 REM
00285 A0$(C)=A0$(C)+K1$+"|": RETURN  REM "|" delimits section
00286 K1$="": X=SEARCH(K0$,"|"): IF X=0 THEN RETURN
00287 K1$=K0$(;1,X-1): K0$=K0$(;X+1): IF Q>0 THEN LET X=X-1
00288 RETURN
00289 REM
00290 REM  ---------- Extract Date Variables Sub. ----------
00291 REM
00292 K1$=A0$(X): F1=VAL(K1$): Y=SEARCH(K1$,"/")+1: K1$=K1$(;Y)
00293 F2=VAL(K1$): Y=SEARCH(K1$,"/")+1: K1$=K1$(;Y)
00294 F3=VAL(K1$): F4=100*F3 + F2 + F1/100: RETURN
00295 REM
00296 REM   ---------- Print Screen heading Sub. -----------
00297 REM
00298 CLS
00299 CURS 17,1: INVERSE: PRINT "^*^Dreamcards^LedgerMaster^*^"
00300 NORMAL: PRINT [A64^45];: RETURN
00301 REM
00302 REM  ------ ---------- Print BASE Sub. ----------------
00303 REM
00304 GOSUB 298: PRINT "PAYMENT^CATEGORIES:^^(": B0$(0); ")": K2$="Category"
00305 IF P=0 THEN RETURN  REM Payments
00306 FOR X=1 TO P: K0$=B0$(X): F1=B2(X): GOSUB 310: NEXT X: RETURN
00307 GOSUB 298: PRINT "RECEIPTS^CATEGORIES:^^(": B0$(0); ")"
00308 IF R=0 THEN RETURN  REM Receipts
00309 FOR X=1 TO R: K0$=B1$(X): F1=B3(X): GOSUB 310: NEXT X: RETURN
00310 J=X+3: K=3: IF X>10 THEN LET J=J-10: K=35
00311 CURS K,J: PRINT CHR$(X+64); ".^"; K0$; SPC(13-LEN(K0$));
00312 IF F1>0 THEN PRINT [F10.2^F1]  REM Print any balance
00313 RETURN
00314 REM
00315 REM -------------- "Counter" Print Sub. --------------
00316 REM
00317 IF X>1 THEN 319
00318 CURS 1,12: PRINT [A26^32] "*^WAIT^*" [A92^32]
00319 IF X>0 THEN CURS 28,13: PRINT [I4^X];
00320 CURS 0: RETURN
00321 REM
00322 REM -------------- Tape I/O Prompt Sub. --------------
00323 REM
00324 K2$="Base^(1)": IF N>2 THEN LET K2$=K2$+",^Files^(2),^All^(3)"
00325 IF N>3 THEN LET K2$=K2$+",^Merge^(4)"
00326 GOSUB 226: H=INT(VAL(K1$)): IF H>N THEN LET H=0
00327 O=H: CURS 26,12: PRINT "Start^tape": CURS 0: RETURN
00328 REM
00329 REM ------------- Initialise Arrays Sub. -------------
00330 REM
00331 IF C=0 THEN 333  REM Print prompt if existing file
00332 GOSUB 232: IF K1$="N" THEN RETURN
00333 CLEAR: V=320: STRS(V*51+1400)  REM V = Max. No. of files
00334 DIM A0(V+2), A1(V+2), B0(20), B1(20), B2(20)
00335 DIM B3(20), B4(20), B5(20): GOSUB 337: GOSUB 338
00336 FOR X=C TO V+2: A0$(X)="": NEXT X: K1$="": RETURN
00337 FOR X=Y TO 20: B0$(X)="": B2(X)=0: NEXT X: RETURN
00338 FOR X=Y TO 20: B1$(X)="": B3(X)=0: NEXT X: RETURN
```

NOTES:

Son of LEDGERMASTER



Ledgermaster is a home and small-business accounting program for tape-based Microbee computers, which first appeared in the May 1985 issue of *Your Computer*.

Since then I've been deluged with letters from Microbee owners who want to get Ledgermaster up and running on their disk machines, and from people who own other brands (poor souls) and want to convert it. As Dame Edna Everage would say, "How can I ignore my public?" So, after lots of re-programming, I'm pleased to announce the birth of 'Son of Ledgermaster', a new version designed for disk machines. Instructions for conversion to other BASICs are outlined below, for those who want to run Ledgermaster on inferior machines, such as the IBM and the Apple.

You may be wondering why we have two versions of Ledgermaster in this magazine. The original financial database program was written for cassette-based Microbees, but enough people found it useful for us to publish a disk-based version. You'll still have to do an awful lot of typing, though . . .

Operating Instructions

Ledgermaster enables you to create, manipulate, link or print disk ledgers, each of which comprises a base and up to 320 files or transactions.

The ledger base is a list of up to 20 categories of payments (for example, rent and insurance), up to 20 categories of receipts (such as wages and interest), and any 'carried forward' totals you want to include. It's extremely important, since it forms the nucleus of the final balance sheet.

When you run Ledgermaster, a menu gives you various options, depending on whether or not a ledger is present in memory. As detailed instructions were given in the earlier article, I won't go into them in depth here. The following is only a summary of the main points.

Create: Allows entry of the base categories of a new ledger.

Enter: Allows you to enter a transaction, requesting first the date (note error checking), then the name of the person who was paid (or who paid you), the cheque or receipt number (if any), whether the transaction is a receipt or payment and its category, and finally the amount. All this data is stored in the file created for the transaction.

Save: You can save a full ledger to disk or, if you want to 'cut and paste', save the base or the files alone. Unlike the tape version, ▶

Ledgermaster II will ask you to specify the drive and filename under which the ledger is to be stored.

Load: You can load a full ledger from disk, or the base or files on their own, or merge the files from a disk ledger with those of the ledger currently in memory.

Sort: Sorts the ledger files in chronological order.

Delete: Deletes a single file or a block of files from a ledger.

New Base: Adds the amount of each file to the 'carried forward' total of the relevant base category, then erases all files. this allows you to 'chain' ledgers by starting a new one where the old one finished.

Print: Ledgers can be displayed on the screen or output to a printer. If a Search string is specified, only files containing that string will be printed. If the search string is a hash mark and a number, such as #99, only those files after the specified file number will be printed. If no search string is set, the printout will comprise three parts: a list of the base categories, a printout of all ledger files and a profit and loss statement. This is quite acceptable for business or taxation purposes.

Entering The Program

The listing shown below will not run, since memory use is too critical to allow REMs and spaces between commands. Enter it exactly as shown, but omit lines containing only a REM and REMs at the end of program lines. Note that variables I, O and Q are listed in lower case so they won't be confused with 1s and 0s.

Save this first draft of the program on disk. The next step is to use GX to delete all spaces in the program (GX/ //). This will take some time! Save the second draft, replace all up-arrows with spaces (GX/↑/ /), and re-number the program to begin at line 1 with one-line increments (RENUM 1,1). You should now have a fully operational Ledgermaster.

Don't despair if the thought of typing a program this size gives you the horrors! Send a cheque or postal order for $20 to Dreamcards, 8 Highland Court, Eltham North, 3095 (mail order only — remember to include a note with your return address!), and I'll send you a tape you can load to disk with TDCOPY, or with the BASIC load and save commands (sorry, I don't have access to a 13 cm machine). If you have a Computer-in-a-Book system, you can send me a formatted disk and a cheque for $18, and I'll copy the program for you. Make sure you pack it extremely well — disks sent through the mail tend to arrive in pieces!

Conversion to Other BASICs

If your machine has less than 32 Kbytes of user RAM, a simple (for example 8 to 12 Kbytes) BASIC, or a screen width of less than 60 characters, forget it! Such a computer is really only suited to video games, and you'd be better off upgrading. But if your machine doesn't fall foul of these minimum requirements, and you have at least a passing acquaintance with BASIC programming, conversion shouldn't be a major task. Assuming not everyone will feel up to it, I should also make it clear that I don't object to user groups or individuals distributing copies — provided my authorship is acknowledged. Otherwise, my team of trained lawyers and Mafia hit-men may descend on you from a very great height!

The main conversion hassle you may have is the odd syntax of Microworld BASIC, particularly in its use of variables, sub-string extraction, print formatting and disk commands.

Variable Conventions

Microworld BASIC doesn't allow you to assign any old name to a variable. Integer or whole-number variables must be designated by a single letter, and cannot take part in real-number maths without special conversion functions. Real-number variables are designated by a letter and a number, such as F2, and can't take part in integer maths. Strings are designated in the same way as real-number variables, but are followed by the conventional '$' sign. These rules impose some minor limitations on Microbee owners, but shouldn't cause problems in most other BASICs.

Variable types may be converted with the INT, FLT, FRACT, STR and VAL commands. INT converts reals to integers by ignoring the fractional part (for example, Z=INT(F1) sets Z=18, where F1=18.7), and FLT converts integers to reals (F1=FLT(Z)/2 sets F1=2.5, where Z=5). FRACT sets a real to the fractional part of another real, thus F1=FRACT(FLT(Z)/2) leaves F1=0.5 where Z=5. STR$ transforms an integer or real into a string (A0$=STR$(X) leaves A0$ containing '123' if X=123). The leading space in the string is reserved for any minus sign.

Variables may be DIMensioned into arrays, although string arrays must first be dimensioned as reals and then converted (as in line 360). The CLEAR command is used to erase all existing array dimensions.

Assuming not everyone will feel up to it, I should also make it clear that I don't object to user groups or individuals distributing copies — provided my authorship is acknowledged.

Sub-String Extraction

Microworld BASIC doesn't use LEFT$, RIGHT$ or MID$, but rather the command v$(;s,e), where v$ is the string from which the sub-string is to be taken, 's' is an integer denoting the first letter to be included, and 'e' is an integer denoting the last. So, if A0$=A1$(;4,X), A1$='hippopotamus' and X=6, A0$ is set to 'pop'. Where the command includes only a single integer or integer expression, all letters from the one designated by the integer to the end of the string will be extracted (in our last example, if A0$=A1$(;6), A0$ would contain 'potamus').

The search command sets an integer to the position within a string of a particular sub-string, taking the format i=SEARCH(a$,b$,n), where 'i' is the integer variable to be set to the search result, a$ is the string to be searched, b$ is the string you're searching for and 'n' is an integer which specifies that you're looking for the 'nth' occurrence of b$. If 'n' is omitted, the command assumes you're looking for the first occurrence. For example, if C=SEARCH(A1$,A0$), A1$='hippopotamus' and A0$='pot', C would equal 6 (the sub-string was found starting at the 6th letter of A1$). If A0$='cat' the search result (C) would equal 0, indicating that the sub-string wasn't found. If A0$='p' and we used B=SEARCH(A1$,A0$,3), B would equal 6, since the third occurrence of the sub-string starts at letter 6 of A1$.

If your BASIC has no search command, you can synthesise it with a suitable subroutine. If we use variable X as the search result, X0$ as the string under search and X1$ as the sub-string, the subroutine will be as follows: ▶

```
1000 REM  'SEARCH' replacement Subroutine
1010 W=1: X=0: Y=LEN(X1$)-1: Z=LEN(X0$)-Y
1020 IF Z<0 THEN RETURN
1030 X2$=X0$(;W,W+Y): REM ie: X2$=MID$(X0$,W,W+Y)
1040 IF X2$=X1$ THEN LET X=W: RETURN
1050 W=W+1: IF W<=Y THEN 1030
1060 RETURN
```

Print Formatting
Ledgermaster uses a 64 by 16 screen, though there is no reason why a larger (say 80 by 24) format wouldn't work just as well. The display/print commands you'll encounter in the program are listed below.
CLS: Clears the screen and positions the cursor at top left.
CURS a,b: Positions the cursor 'a' (1 to 64) characters to the right of screen and 'b' (1 to 16) lines down.
CURS 0: Turns off the flashing cursor until the next PRINT command.

NOTES:

PRINT: Prints a carriage return and line feed. It's the equivalent of PRINT CHR$(13) CHR$(10).
PRINT SPC(n): Prints 'n' spaces — like PRINT TAB(n) PRINT [An c]. The square brackets designate a 'PRINT USING' command and the 'A' denotes ASCII text. This version prints the character that is the ASCII equivalent of integer 'c', 'n' times. It's the same as 'FOR X=1 TO n: PRINT CHR$(c);: NEXT X'.
PRINT [Iv n]: Prints the integer variable 'v' right-justified in a field 'n' characters wide (only used in line 343).
PRINT [Fn.m r]: Prints the real variable 'r' right-justified in a field 'n' characters wide. Limit the real number to 'm' decimal places.
OUT#n: A Microbee curiosity used to redirect print output (for example, lines 166 and 216, where variable q designates that print is to go to the screen if q=0, or to a printer if q>0). OUT#0 and IN#0 are used at various points in the program to turn the screen back on where it has been disabled by some other command.
Disk Commands: Microworld BASIC uses the OPEN command to open a disk file for sequential read or write operations. The format is:

OPEN "f",6,"d:FILENAME.EXT"

where 'f' is the function ('I'=READ, 'O'=WRITE), the '6' is a port vector allocation which can be ignored in conversion, 'd' is the drive code ('A', 'B' or 'C'), and the FILENAME and EXTENSION are as in CP/M (maximum of eight and three characters respectively). A string variable may be substituted for any of these attributes.

After a file has been OPENed, a sequence of IN and OUT commands will be used to turn off the screen and re-direct output to the disk port. These commands are irrelevant on other machines. Various other commands, which I've listed below, are used in association with disk operation.
DISKRESET n: This resets the computer's knowledge of the directory of the disk in drive 'n', preventing a BDOS error if the disk is changed before a READ/WRITE operation is performed.
CLOSE 6: This closes the disk file that was opened with the OPEN command. The '6' is a Microbee peculiarity.
ON ERROR GOTO: Microworld BASIC generates an Error Report if you try to OPEN a file which doesn't exist on the particular disk. This command suppresses the report and directs the program to continue at the specified line number if an error occurs.
ON ERROR GOTO 0: Cancels the above.
General Syntax: Some of the general commands used in Microworld BASIC differ a little in other dialects. The ON v GOTO command (see Line 18) jumps to the first line number where v=1, the second where v=2, and so on. Some BASICs jump to the first line number where v=0, and if yours is one of these the command should be amended to ON v-1 GOTO. The STRS(v) command reserves v bytes of memory for the strings, and v=FRE($) sets v equal to the number of free bytes of string space remaining at the time it's executed. PLAY x,y plays a note 'x' (from low frequency=1 to high frequency=22) for y/20 seconds. If x=0 a silent delay is produced. v$=KEY$ reads the keyboard and sets string v$ to any key being pressed at the time. If no key is being pressed, the instruction doesn't wait, but returns a null string (ASCII value 128).

How It Works
Much of Ledgermaster's operation is explained by REMs in the listing, but you'll also need to read these notes to understand it fully. The following are the main program variables ('n' is an integer in the range 0>n<=V).
C: Number of current file being accessed.
H: Disk file type (Base=1, Files=2, Full Ledger=3).
P: Number of 'payment' categories in Base.
R: Number of 'receipt' categories in Base.
S: Files in ledger are Sorted (=1) or Unsorted (=0).
V: Maximum number of files: must be >C.
K5$: Ledger filename (maximum of eight characters).
A0$(n): File string — contains date, payee/or, receipt number, payment/receipt (P or R), category designation letter and ▶

Much of Ledgermaster's operation is explained by REMs in the listing, but you'll also need to read these notes to understand it fully.

amount. Fields are separated by
B0$(n): Payment category.
B1$(n): Receipt category.
B2(n): Payment category amount 'carried forward'.
B3(n): Receipt category amount 'carried forward'.

When the program is first run, the Initialisation routine (accessed from line 2) reserves sufficient string space, clears and resets the arrays, and converts arrays required for strings (A0$, B0$ and B1$) into string arrays. V determines the function of the initialisation routine — the maximum number of files per ledger can be altered simply by changing the value assigned to it in line 357. Crashes will occur if this exceeds 320 on a 32 Kbyte Microbee. Owners of other machines may have to experiment to find an appropriate value.

Lines 3 to 5 then set N to a value which indicates the status of the arrays (or whether a base and/or files are present), and this is used by lines 6 to 14 to determine what options will be displayed on the menu. Line 15 alerts you if the ledger is full. The program then prompts you to select an option, and jumps to the appropriate section when a valid key is pressed (lines 17 to 18).

Note the unusual input subroutine (commencing in line 274). This prints a line of hyphens across the 15th line of the screen, then a prompt (K2$) at left of the 16th line, followed by a number of stars (*), according to the maximum number of characters (G) allowed in the input field. As each character is entered, it replaces one of the stars, the input ending when RETURN is pressed or when the field is full (lines 281, 282 and 284). DELete and BACKSPACE are implemented by lines 285 and 286. A null string may only be entered if I>0 (line 282).

Create Base

Line 68 first asks for a 'File Title' to help identify the file, then stores it in B0$(0) (line 69 — be careful when converting, as some BASICs don't permit an array element numbered 0). If yours is one, I'd suggest you DIM B0$ to 21 elements, use B0$(21) to store the File Title and modify the rest of the program accordingly. The routine goes through two similar loops, the first (Payments) in lines 70 to 73, and the second (Receipts) in lines 75 to 78. These display the respective base categories on each loop via one of two access points in the 'Print Base' subroutine, then ask for a category and an 'Amount Carried'. These inputs are stored in the appropriate arrays. The loop terminates when the base category counter (P or R) reaches 20, or if a null is entered when you're prompted for the category. T is then set to the number of categories having 'Carried Forward' amounts, C is set to 1 to indicate that a base is present, and the program returns to the Menu.

Enter begins at line 86. S is set to 0: the new entry means the ledger is 'unsorted', the File string (A0$(C)) is set to a null, and the 'File Entry' print subroutine is accessed. Since there is nothing in the file, only the file number (C) is displayed. If at least one file is present, line 88 accesses the 'Extract Date Variables' subroutine, and F1, F2 and F3 are set to the date of the last file (see below). Lines 89 to 94 ask for the day, month and year parts of the date, some error checking being performed at each level (for example, 'day' must be >0 and <32). Entries are stored in F1 (day), F2 (month) or F3 (year). A null entry causes the program to jump the remaining date entry lines, leaving the variables holding the values borrowed from the last file by line 88. Lines 95 to 99 check that the day doesn't exceed the number of days in that month and that the 29th of February hasn't been entered on a non-leap year, an error message being generated if an illegal value is detected. Lines 100 to 102 convert the date variables into a string.

Line 103 trims the leading space from the date string, displays it, then asks for a 'From/to' input. If this is a null, 'Paid' is inserted into the file by line 104. Line 105 displays the file so far and asks for a receipt or cheque number, and line 106 inserts 'B'card' into the file if ↑B (CTRL B) was pressed, or 'Cash' if ↑A was pressed. Line 107 displays the file, asks if it's a receipt or payment ('R' or 'P' is stored). Lines 109-113 then update the file, display the Base Receipt or Payment categories (as appropriate), and ask for one to be selected. The letter code is stored and you are asked for the 'Amount' (line 115 — error signalled by X=1). Lines 116 to 119 permit the current file to be re-typed, the next file to be entered, or a return to the menu.

Delete starts at line 142, asking for the number of the (first) file to be deleted (o). Invalid file numbers are rejected, otherwise the file is displayed, and line 144 asks if it's the one you want. If the answer is 'N' the program returns to the menu, otherwise line 145 asks for the number (L) of the last file to be deleted (displayed by line 148), or (RETURN) if only one file is involved. Lines 149 to 150 make the deletion, condensing the arrays to allow for the missing file/s.

Sort starts at line 22 (low in the program to enhance its speed), asks if you're sure, and refuses to operate if the ledger is already sorted (S=1). Line 24 then converts the date part of each file into a number (F4) and stores it in the A1 array, then lines 25 to 29 carry out an insertion sort of the file variables. BASIC sorts are extremely slow, so a counter is displayed as each array element is selected. S=1 on completion.

New Base begins at line 222. This line asks if you're sure and line 223 asks for a new File Title and stores it in B0$(0) (see 'Create Base'). A null input retains the old File Title. Lines 224 to 225 begin a loop which extracts the 'P' or 'R' string from each file (O is set to its ASCII value), the Category string (Z is set to the appropriate number from 1 to 20), and the Amount string (converted to a real in F1). Line 226 adds F1 to the the appropriate element in either the B2 or B3 array, and the loop continues. Line 227 calls the subroutine at line 360, and returns to the menu.

Print starts at line 154, asking if you want any 'Search String' (K4$). If K4$ begins with a hash mark (#), line 155 sets H to the value of any following number (line 156 rejects invalid file numbers). Line 157 asks if you're using the VDU or a printer. If it's a printer, lines 159 to 164 allow you to select the type. During these routines Q is set to the output mode (0 if VDU, >0 if printer), and L is set to the number of lines per page (VDU=12, Printer=57). If printer output was selected, line 166 asks for a key, then turns the VDU off and the printer on. Line 167 copies the Base 'Carried Forward' arrays (B2 and B3) into the 'Ledger Balance' arrays (B4 and B5). Lines 168 to 169 set some of the print variables, E being the total number of lines in the file, O the ▶

Sort starts at line 22 (low in the program to enhance its speed), asks if you're 'Sure', and refuses to operate if the ledger is already sorted (S=1).

number of lines occupied by the base categories, and U and W being counters.

The print routine loop begins at line 170, lines 170 to 172 detecting any key pressed, aborting if it's 'A', or looping until another key is pressed if not. Lines 173 to 183 use the line counter (D) to detect top of page, and print a page heading if a printer is used or clear a section of screen if the VDU is used. If at the top of the listing (o=0) and if no search string is defined, the Base categories are printed out by lines 184 to 194, otherwise line 195 prints what is being searched for. Lines 196 to 197 print the files themselves by accessing the 'File Entry' print subroutine (fully detailed by REMs in the listing), then lines 198 to 199 add the Amount to the appropriate Balance array element. Lines 200 to 210 are invoked once the files are printed, and are responsible for printing the final balance.

Load and Save are detailed by REMs in the listings and the descriptions of the main variables (above). O=Command (Cmnd.) type (files only, full ledger, and so on) and H=disk file type. Ledger files are saved or loaded three at a time, since it's the most efficient way to use the Microbee cassette system. Although not necessary for disk use, it's been retained to maintain consistency with the cassette version.

Dead Bugs

Two bugs have been corrected in this version. When entering a file cheque/receipt number a ↑ C entry was intended to enter 'Cash'. This didn't work because the BREAK key was disabled, hence it's been changed to ↑ A (line 106). A new line (48) has been inserted in the Load Routine to overcome a problem which occurred when 'Base' alone was loaded. □

```
00001 POKE 140,1  REM Disable BREAK key
00002 GOSUB 357  REM Initialise Storage Arrays
00003 IN#0: OUT#0  REM VDU On - Main program loop begins here
00004 N=8: IF B0$(0)="" THEN LET N=2 ELSE IF C=1 THEN LET N=4
00005 K0$=KEY$: K4$="": i=0: q=0: IF C<3 AND N=8 THEN LET N=5
00006 GOSUB 322: INVERSE: CURS 28,4: PRINT "^MENU^": NORMAL
00007 PRINT \ SPC(14) "Create^.....^1";
00008 PRINT SPC(4) "Load^.......^2": IF N=2 THEN 16  REM No Base
00009 PRINT SPC(14) "Enter^......^3";
00010 PRINT SPC(4) "Save^.......^4": IF N=4 THEN 16 REM No Files
00011 PRINT SPC(14) "Delete^.....^5";: IF N=5 THEN 16 REM <3 Files
00012 PRINT SPC(4) "Sort^.......^6"
00013 PRINT SPC(14) "Print^......^7";
00014 PRINT SPC(4) "New^Base^...^8"
00015 IF C=V+1 THEN PRINT \ SPC(21) "***^LEDGER^FULL^***"
00016 K2$="Select^Menu^Option": GOSUB 246  REM Get key
00017 X=INT(VAL(K0$)): IF X=0 OR X>N THEN 16  REM Check key range
00018 PLAY 22,1: ON X GOTO 68,33,86,123,142,22,154,222
00019 REM
00020 REM ------------------ Sort Routine ----------------------
00021 REM
00022 GOSUB 252: IF K1$="N" OR S=1 THEN 3 REM "No" or already sorted?
00023 X=0: GOSUB 341  REM Set A1 Array to file dates
00024 FOR X=1 TO C-1: GOSUB 316: A1(X)=F4: NEXT X
00025 FOR X=1 TO C-2: GOSUB 341: FOR M=X+1 TO C-1  REM Sort
00026 IF A1(M)=>A1(X) THEN 29  REM Later or same date?
00027 K0$=A0$(M): A0$(M)=A0$(X): A0$(X)=K0$ REM Substitute files
00028 F1=A1(M): A1(M)=A1(X): A1(X)=F1 REM Substitute sort dates
00029 NEXT M: NEXT X: S=1: GOTO 82
00030 REM
00031 REM ---------------------- Load ---------------------------
00032 REM
00033 N=3: IF C>1 AND C<V+1 THEN LET N=4  REM Merge available?
00034 K5$="LOAD": GOSUB 231: IF K2$="" THEN 82  REM Get Filename
00035 ON ERROR GOTO 239: OPEN "I",6,K5$+".LGR": CLOSE 6  REM Valid?
00036 ON ERROR GOTO 0: GOSUB 348: IF o=0 THEN DISKRESET "A": GOTO 3
00037 OPEN "I",6,K5$+".LGR": IN#6ON: OUT#0: OUT#0OFF  REM Get disk file
00038 INPUT K1$,D,H,X,Y,Z,E  REM Input File Header
00039 IN#0: OUT#0  REM VDU back on
00040 IF o=3 AND H<3 THEN LET o=0 REM Cmnd. = Ledg., Disk=Files/Base
00041 IF o=1 AND H=2 THEN LET o=0  REM Cmnd.=Base, Disk=Files
00042 IF o>1 AND H=1 THEN LET o=0  REM Cmnd.=Files/Ledg., Disk=Base
00043 K0$="": J=1: L=D-1: IF o<4 THEN 46  REM Goto 46 if not Merge
00044 J=C: L=L+C: C=L: S=0: K0$="^-^Merge"
00045 IF C>V+1 THEN LET C=V+1: K0$=K0$+"^(Part^only)"
00046 IF o=2 OR o=3 THEN LET C=D: S=Z  REM Files/Ledg.
00047 IF o=1 OR o=3 THEN LET B0$(0)=K1$: P=X: R=Y: T=E  REM Base/Ledg.
00048 IF o=1 AND C=0 THEN LET C=1  REM Corrects bug in tape version
00049 K1$=K1$+K0$: CURS 0,12: PRINT [A64^32]  REM Display File Title
00050 CURS (62-LEN(K1$))/2,12: PRINT K1$: CURS 26,13
00051 IF H=1 OR o=1 OR C=1 THEN PRINT "^^(Base)": GOTO 54
00052 IF (H=2 OR o=2) AND C>1 THEN PRINT "^^(File)": GOTO 54
00053 IF S=0 AND C>1 AND o>1 THEN PRINT "(Unsorted)"
00054 IF o=0 THEN 82 ELSE LET N=X: IF Y>X THEN LET N=Y
00055 CURS 0: PLAY 0,10: IN#6ON: OUT#0: OUT#0OFF  REM VDU Off, Disk On
00056 FOR X=1 TO N  REM Load Base
00057 IF (o=2 OR o=4) AND H=3 THEN INPUT K0$,K0$,F1,F1  REM Dummy load
00058 IF o=1 OR o=3 THEN INPUT B0$(X),B1$(X),B2(X),B3(X) REM Real load
00059 NEXT X: IF o=1 THEN 61 ELSE FOR X=J TO C STEP 3  REM Load Files
00060 INPUT A0$(X),A0$(X+1),A0$(X+2): NEXT X
00061 IF C<V+1 AND o>1 THEN GOSUB 360  REM These lines clear
00062 Y=P+1: IF P<20 THEN GOSUB 361    REM irrelevant array
00063 Y=R+1: IF R<20 THEN GOSUB 362    REM elements.
00064 IN#0: OUT#0: CLOSE 6: DISKRESET "A": GOTO 82 REM Tidy up and end
00065 REM
00066 REM ----------------- "Create" Routine ------------------  ▶
00067 REM
```

```
00068 GOSUB 355: IF K1$="N" THEN 3  REM Are you sure?
00069 G=24: K2$="File^Title": GOSUB 322: GOSUB 274: B0$(0)=K1$: i=1
00070 GOSUB 328: P=P+1: GOSUB 269: B0$(P)=K1$: K3$=K1$ REM Payments
00071 IF K3$="" AND P=1 THEN LET B0$(1)="Payment": GOTO 73
00072 IF K3$="" THEN LET P=P-1: GOTO 74
00073 GOSUB 257: B2(P)=F3: IF P<20 AND K3$<>"" THEN 70
00074 PLAY 22,1
00075 GOSUB 331: R=R+1: GOSUB 269: B1$(R)=K1$: K3$=K1$ REM Receits
00076 IF K3$="" AND R=1 THEN LET B1$(1)="Receipt": GOTO 78
00077 IF K3$="" THEN LET R=R-1: GOTO 79
00078 GOSUB 257: B3(R)=F3: IF R<20 AND K3$<>"" THEN 75
00079 PLAY 22,1: T=0: FOR X=1 TO 20: IF B2(X)>0 THEN LET T=T+1
00080 IF B3(X)>0 THEN LET T=T+1 REM Count no. of Carried Forward's
00081 NEXT X: C=1: K3$="": GOTO 3
00082 CURS 0: PLAY 22,1; 0,5: GOTO 3
00083 REM
00084 REM ----------------- "Enter" Routine --------------------
00085 REM
00086 IF C>V THEN 3  REM No more entries if ledger full
00087 S=0: A0$(C)="": M=C: GOSUB 292: i=C-1: G=2 REM Display file
00088 IF C>1 THEN LET X=C-1: GOSUB 316  REM Get last file date
00089 K2$="Day": GOSUB 274: IF K1$="" AND F1>0 THEN 95
00090 F1=VAL(K1$): IF F1=0 OR F1>31 THEN 89 REM F1=Day
00091 K2$="Month": GOSUB 274: IF K1$="" AND F2>0 THEN 95
00092 F2=VAL(K1$): IF F2=0 OR F2>12 THEN 91 REM F2=Month
00093 K2$="Year": GOSUB 274: IF K1$="" AND F3>0 THEN 95
00094 F3=VAL(K1$): IF F3=0 THEN 93 REM F3=Year
00095 RESTORE: FOR X=1 TO INT(F2): READ Z: NEXT X REM Check days
00096 DATA 31,28,31,30,31,30,31,31,30,31,30,31    REM in month and
00097 IF FRACT(F3/4)=0 AND F2=2 THEN LET Z=29     REM Allow for Feb.
00098 IF INT(F1)<=Z THEN 100  REM Invalid no. of days in month
00099 CURS 1,16: PRINT "***^DATE^ERROR^***";: PLAY 10,12: GOTO 89
00100 K0$=STR$(INT(F1)): K1$=K0$(;2)+"/"  REM Form date string
00101 K0$=STR$(INT(F2)): K1$=K1$+K0$(;2)+"/"
00102 K0$=STR$(INT(F3)): IF F3<10 THEN LET K1$=K1$+"0"
00103 K1$=K1$+K0$(;2): GOSUB 291: K2$="From/To": G=16: i=1
00104 GOSUB 274: IF K1$="" THEN LET K1$="Paid"
00105 GOSUB 291: K2$="Cheque/Receipt^No.": G=7: GOSUB 274
00106 IF X=2 THEN LET K1$="B'card" ELSE IF X=1 THEN LET K1$="Cash"
00107 GOSUB 291: K2$="Receipt^(R)^or^Payment^(P)"
00108 GOSUB 246: IF K1$<>"R" AND K1$<>"P" THEN 108
00109 GOSUB 309: GOSUB 322
00110 IF K1$<>"P" THEN 112
00111 M=P: IF M=1 THEN LET K1$="A": GOTO 114 ELSE GOSUB 328: GOTO 113
00112 M=R: IF M=1 THEN LET K1$="A": GOTO 114 ELSE GOSUB 331
00113 K2$="Category": GOSUB 246:X=X-64: IF X<1 OR X>M THEN 113
00114 GOSUB 291
00115 G=9: i=0: K2$="Amount": GOSUB 274: GOSUB 262: IF X=1 THEN 115
00116 GOSUB 291: K2$="Next=<CR>^^Menu=M^^Re-do=R^": i=1
00117 GOSUB 246: IF K1$="R" THEN 87  REM Re-do?
00118 IF K1$<>"M" AND X<>128 THEN 117
00119 C=C+1: IF X=128 THEN 86 ELSE 3  REM Update file counter & loop
00120 REM
00121 REM ------------------ "Save" Routine --------------------
00122 REM
00123 N=3: IF C=1 THEN LET N=1  REM If C=1 then allow Base Save only
00124 K5$="SAVE": GOSUB 231: IF K2$="" THEN 82  REM Drive/Filename
00125 q=0: ON ERROR GOTO 126: OPEN "I",6,K5$+".LGR": q=1 REM New File?
00126 ON ERROR GOTO 0: IN#0: OUT#0: CLOSE 6: IF q=0 THEN 129
00127 K2$="OVERWRITE^"+K5$+"^(Y/N)": GOSUB 246 REM Filename in use
00128 IF K1$="N" THEN 124 ELSE IF K1$<>"Y" THEN 127
00129 GOSUB 348: IF H=0 THEN DISKRESET "A": GOTO 3  REM Menu if invali
00130 K0$=B0$(0): N=P: IF R>P THEN LET N=R REM is Base array counter
00131 CURS 0,12: PRINT [A24^32]; "Saving^"; K5$; [A33^32]: CURS 0
00132 OPEN "O",6,K5$+".LGR": OUT#6  REM Next line saves Header
00133 PRINT "'";K0$;"',";C;",";H;",";P;",";R;",";S;",";T
00134 IF H=2 THEN 136 ELSE FOR X=1 TO N  REM Save Base arrays
```

```
00135 PRINT B0$(X);",";B1$(X);",";B2(X);",";B3(X): NEXT X
00136 IF C<2 OR H=1 THEN 64  REM End if Base only
00137 FOR X=1 TO C STEP 3  REM Save File array
00138 PRINT A0$(X);",";A0$(X+1);",";A0$(X+2): NEXT X: GOTO 64
00139 REM
00140 REM ----------------- "Delete" Routine -------------------
00141 REM
00142 G=3: K3$="^file^to^be^deleted": K2$="First"+K3$ REM Get first
00143 GOSUB 274: o=INT(VAL(K1$)): IF o<1 OR o=>C THEN 142 REM Valid?
00144 M=o: GOSUB 292: GOSUB 252: IF K1$="N" THEN 3 REM Display file
00145 G=3: K2$="Last"+K3$+"^-^or^<CR>": i=1: GOSUB 274 REM Get last
00146 L=INT(VAL(K1$)): IF K1$="" THEN LET L=o REM Single file only
00147 i=0: IF L<o OR L=>C THEN 145 ELSE IF L=o THEN 149 REM Valid?
00148 M=L: GOSUB 292: GOSUB 252: IF K1$="N" THEN 145 REM Display file
00149 K3$="": Z=L-o+1: X=0: GOSUB 341: FOR X=L+1 TO C REMAdjust files
00150 A0$(o)=A0$(X): o=o+1: NEXT X: C=C-Z: GOSUB 360: GOTO 82
00151 REM
00152 REM ----------------- "Print" Routine --------------------
00153 REM
00154 G=20: i=1: H=1: K2$="String^search": GOSUB 274: K4$=K1$
00155 IF ASC(K4$)=35 THEN LET H=INT(VAL(K4$(;2))): K4$="#"
00156 IF H=0 OR H>C-2 THEN 154 REM Loop if invalid number in K4$
00157 K2$="Screen^(S)^or^Printer^(P)": GOSUB 246
00158 L=12: IF K1$="S" THEN 167 ELSE IF K1$<>"P" THEN 3
00159 GOSUB 322: K2$="PRINTER^Type:^": PRINT \\ SPC(15); K2$
00160 PRINT \ SPC(20) "Parallel^^^^^^-^^Key^1"  REM q is MicroBee
00161 PRINT SPC(20) "Serial^300bd^^-^^Key^2"    REM Printer port
00162 PRINT SPC(20) "Serial^1200bd^-^^Key^3"    REM allocation
00163 PRINT SPC(20) "Abort^-^^any^other^Key"    REM vector.
00164 GOSUB 246: X=X-48: IF X<0 OR X>4 THEN 3
00165 L=57: q=X: IF X>1 THEN LET q=X+2  REM Adjust to port number
00166 GOSUB 245: OUT#0: OUT#0OFF: OUT#qON: PRINT  REM Printer on
00167 FOR X=1 TO 20: B4(X)=B2(X): B5(X)=B3(X): NEXT X REM Balance var's
00168 D=0: F3=0: F4=1: o=T: U=0: W=0: E=C+P+R+T+4
00169 IF K4$<>"" THEN LET E=C: o=1  REM If Search String defined
00170 G=1: K0$=KEY$: IF K0$="" THEN 173 ELSE PLAY 22,2  REM Key?
00171 IF K0$="A" OR K0$="a" THEN 216  REM Abort if Key <A>
00172 K1$=KEY$: IF K1$="" THEN 172  REM Wait for another key
00173 IF D>0 THEN 184 ELSE IF q>0 THEN 177  REM D is line counter
00174 REM Next line clears screen if D=0 & not using printer
00175 FOR X=3 TO 14: CURS1,X: PRINT [A64^32]: NEXT X: CURS1,3: GOTO 184
00176 REM Next 7 lines print page heading if D=0 & using printer
00177 D=2: PRINT "LEDGER:^^"; B0$(0); SPC(26-LEN(B0$(0)));
00178 K2$="^to^": IF S=0 THEN 181  REM Sorted?
00179 K0$=A0$(1): GOSUB 310: K2$=K1$+K2$ REM Print File#1 date
00180 K0$=A0$(C-1): GOSUB 310: K2$=K2$+K1$: PRINT K2$; REM Print last
00181 PRINT SPC(34-LEN(K2$)); "Page^"; F4: F4=F4+1 REM Print Page No.
00182 PRINT "No.^^^^^DATE^^^^TRANSACTION"; SPC(17); REM Print heading
00183 PRINT "CLASS"; SPC(16); "AMOUNT^($)": PRINT [A7^45]
00184 IF o=0 OR K4$<>"" THEN 194 REM If Search String or not at top
00185 U=U+1: IF U>P THEN 188 REM Base Cat's (Payments) printed?
00186 F1=B2(U): IF F1=0 THEN 185 REM Any Carried Forward?
00187 K0$=B0$(U): X=LEN(K0$): Z=0: GOTO 190
00188 W=W+1: F1=B3(W): IF F1=0 THEN 188 REM Receipts
00189 K0$=B1$(W): X=LEN(K0$): Z=11
00190 IF o=T THEN PRINT "^CARRIED^FORWARD:^^^"; ELSE PRINT SPC(20);
00191 IF q>0 THEN PRINT SPC(22); K0$; SPC(27-X-Z);
00192 IF q=0 THEN PRINT K0$; SPC(33-X-Z);
00193 o=o-1: PRINT [F10.2^F1]: GOTO 211  REM Print amount carried
00194 IF K4$="" OR K4$="#" OR o<>1 THEN 196
00195 PRINT \ SPC(10); "SEARCH:^"; K4$ \: o=0: G=3: GOTO 211
00196 M=H: IF K4$="" THEN LET M=H-T
00197 IF M<1 OR M>=C THEN 200 ELSE GOSUB 293  REM Print file
00198 IF M=0 THEN LET B5(N)=B5(N)+F1 ELSE LET B4(N)=B4(N)+F1
00199 GOTO 211  REM Add balance in file to BASE category
00200 IF K4$<>"" THEN 211  REM Goto 211 if Search string set
00201 M=M-C: IF M=0 THEN PRINT: GOTO 211  REM File End
```

▶

NOTES:

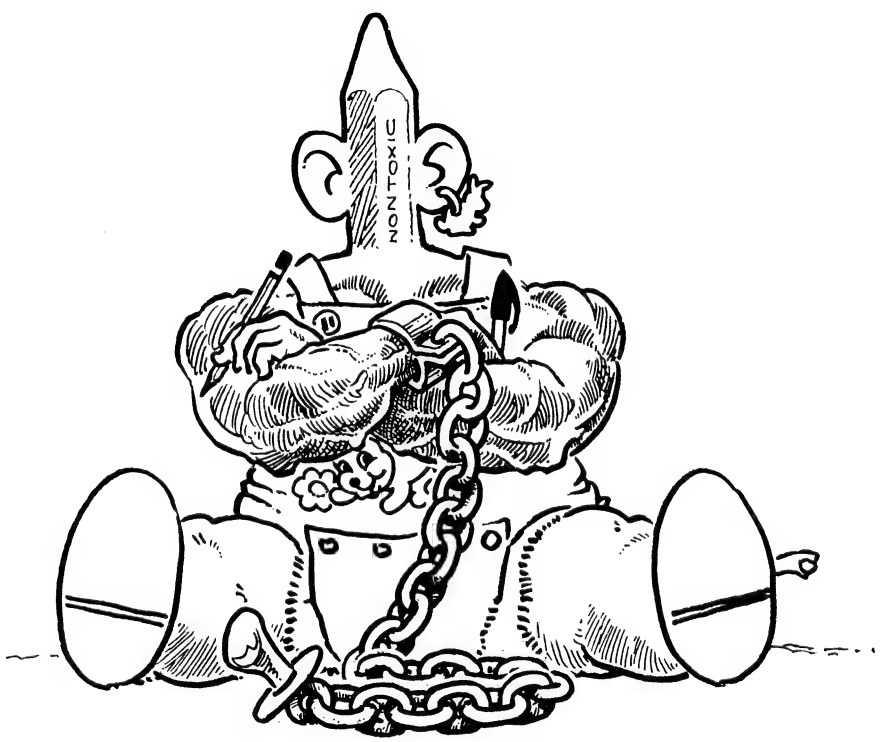
```
00202 IF M>P THEN 205 ELSE IF B4(M)=0 THEN 215  REM Payments
00203 PRINT SPC(20); B0$(M); SPC(22-LEN(B0$(M))); [F10.2^B4(M)]
00204 F3=F3-B4(M): GOTO 211
00205 M=M-P: IF M>R THEN 208 ELSE IF B5(M)=0 THEN 215  REM Receipts
00206 PRINT SPC(20); B1$(M); SPC(22-LEN(B1$(M))); [F10.2^B5(M)]
00207 F3=F3+B5(M): GOTO 211
00208 X=67: IF q=0 THEN LET X=51  REM Print total at end
00209 IF H=E OR H=E-2 THEN PRINT SPC(X+2); [A10^45]: G=1
00210 IF H=E-1 THEN PRINT SPC(X); [F11.2^F3]: G=1
00211 D=D+G: IF D<L THEN 215  REM L is Lines per page counter
00212 REM Delete next line if not using fan-fold paper
00213 IF q>0 THEN PRINT \\\\\\\  REM Perforation gap = 7 lines
00214 D=0: IF q>0 THEN 215 ELSE GOSUB 245: IF K0$="A" THEN 218
00215 H=H+1: IF H<=E THEN 170  REM Detect end of print run
00216 IF q>0 THEN OUT#qOFF: GOTO 218
00217 IF K0$<>"A" AND K0$<>"a" THEN GOSUB 245
00218 q=0: GOTO 3
00219 REM
00220 REM --------------- "New Base" Routine --------------------
00221 REM
00222 GOSUB 252: IF K1$="N" THEN 3  REM Are you sure?
00223 i=1:G=24:K2$="New^Title":GOSUB 274: IF K1$<>"" THEN LET B0$(0)=K1$
00224 FOR Y=1 TO C-1: K0$=A0$(Y): X=SEARCH(K0$,"|",3): GOSUB 311
00225 o=ASC(K0$): GOSUB 310: Z=ASC(K0$)-64: GOSUB 310: F1=VAL(K0$)
00226 IF o=80 THEN LET B2(Z)=B2(Z)+F1 ELSE LET B3(Z)=B3(Z)+F1
00227 X=Y: GOSUB 341: NEXT Y: C=1: GOSUB 360: GOTO 79
00228 REM
00229 REM ---------- Drive/Filename Selection Sub. -------------
00230 REM
00231 POKE 257,1: K2$="Select^Drive?^(A/B/C)": GOSUB 246
00232 IF X<65 OR X>67 THEN LET K2$="": GOTO 235  REM A, B & C only
00233 DISKRESET K1$: K2$=K5$+":^Filename": G=8: GOSUB 274
00234 K5$=K1$  REM Store Filename in K5$
00235 POKE 257,0: RETURN  REM POKE 257 forces Upper/Lower case
00236 REM
00237 REM ----------- Invalid File Reference Message -----------
00238 REM
00239 ON ERROR GOTO 0: IN#0: OUT#0: CLOSE 6: CURS 1,16 REM Close file
00240 PRINT [A63^32];: CURS 22,16: PRINT "<<<^NO^SUCH^FILE^>>>";
00241 CURS 0: PLAY 0,20: DISKRESET "A": K5$="": GOTO 82
00242 REM
00243 REM -------------- "Key to Continue" Sub. ----------------
00244 REM
00245 K2$="KEY^to^Continue"  REM Accessed at Lines 245 & 246
00246 G=1: GOSUB 274: X=ASC(K1$)  REM Next line converts to upper case
00247 IF X>96 AND X<123 THEN LET X=X-32: K1$=CHR$(X)
00248 RETURN
00249 REM
00250 REM ---------------- "Sure (Y/N)" Sub. -------------------
00251 REM
00252 K2$="Sure^(Y/N)": GOSUB 246: IF K1$<>"N" AND K1$<>"Y" THEN 252
00253 RETURN
00254 REM
00255 REM ----------- "Previous Balance" Input Sub. ------------
00256 REM
00257 G=9: K2$="Balance^Forward": GOSUB 274: GOSUB 262
00258 IF X=1 THEN 257 ELSE RETURN  REM Loop if invalid amount
00259 REM
00260 REM ---------------- "Money Input" Sub. ------------------
00261 REM
00262 X=1: F1=VAL(K1$): IF F1<=0 AND i=0 OR F1=>1000000 THEN 265
00263 X=0: F2=FRACT(F1): F3=F1-F2+FLT(INT(F2*100))/100
00264 K1$=STR$(F3): K1$=K1$(;2)  REM Round off to 2 decimal places
00265 RETURN
00266 REM
00267 REM ----------------- "Category" Sub. --------------------
00268 REM
```

```
00269 G=12:K2$="Category"
00270 REM
00271 REM ----------------- Main Input Sub. --------------------
00272 REM G = No. of Characters, i = Null allowed, K2$ = Prompt
00273 REM
00274 GOSUB 323: Y=LEN(K2$): CURS 1,15: PRINT [A64^45] [A63^32];
00275 X=V+1-C: IF X>V THEN LET X=V  REM Limit Free Files Counter
00276 CURS 1,16: PRINT K2$;: CURS 52,16: PRINT INT(FRE($)); "^:"; X;
00277 CURS Y+3,16: Z=0: K1$="": FOR X=1 TO G: PRINT "*";: NEXT X
00278 CURS Y+1,16: PRINT "?^";  REM Print Prompt Line
00279 K0$=KEY$: X=ASC(K0$): IF X=44 OR X=124 OR X=128 THEN 279
00280 IF X=8 OR X=127 THEN 285 ELSE IF X>13 THEN 283  REM B/S or DEL
00281 IF Z>0 THEN CURS 0: RETURN  REM End if <CR>
00282 IF i>0 THEN LET K1$="": RETURN ELSE 279
00283 Z=Z+1: CURS Y+Z+2,16: PRINT K0$;  REM Print input so far
00284 K1$=K1$+K0$: IF Z=G THEN RETURN ELSE 279  REM End of string?
00285 K0$=K1$(;1,LEN(K1$)-1): K1$=K0$  REM <DEL>ete key?
00286 IF Z>0 THEN CURS Y+Z+2,16: PRINT "*";: CURS Y+2,16: Z=Z-1
00287 GOTO 279
00288 REM
00289 REM ------------ "File Entry" Print Sub. -----------------
00290 REM
00291 M=C: GOSUB 309  REM M is File number
00292 GOSUB 322: PRINT "FILE:"
00293 IF K4$="" OR K4$="#" THEN 295  REM Implement search if active
00294 K0$=A0$(M): IF SEARCH(K0$,K4$)=0 THEN LET G=0: RETURN
00295 K0$=STR$(M)+".": K0$=K0$(;2): X=LEN(K0$) REM K0$=File No.
00296 PRINT K0$; SPC(4-X);: K0$=A0$(M): GOSUB 310  REM Date
00297 PRINT SPC(9-LEN(K1$)); K1$; "^";: IF q>0 THEN PRINT "^";
00298 GOSUB 310: PRINT K1$; SPC(18-X);  REM Item
00299 GOSUB 310: PRINT K1$; SPC(9-X);  REM Cheque/Receipt No.
00300 GOSUB 310: M=0: IF K1$="P" THEN LET M=11  REM Rec/Pay?
00301 IF q=0 THEN PRINT K1$; "^";  REM Don't print in full on VDU
00302 GOSUB 310: N=ASC(K1$)-64: K2$=B0$(N): IF M=0 THEN LET K2$=B1$(N)
00303 IF q>0 THEN LET X=LEN(K2$): PRINT K2$; SPC(16-X); ELSE PRINT K1$;
00304 GOSUB 310: F1=VAL(K1$): IF F1>0 THEN PRINT SPC(M); [F10.2^F1]
00305 RETURN
00306 REM
00307 REM ----------- Extract File Section Sub's. --------------
00308 REM
00309 A0$(C)=A0$(C)+K1$+"|": RETURN REM "|" delimits sections
00310 K1$="": X=SEARCH(K0$,"|"): IF X=0 THEN RETURN
00311 K1$=K0$(;1,X-1): K0$=K0$(;X+1): IF q>0 THEN LET X=X-1
00312 RETURN
00313 REM
00314 REM ------------ Extract Date Variables Sub. -------------
00315 REM
00316 K1$=A0$(X): F1=VAL(K1$): Y=SEARCH(K1$,"/")+1: K1$=K1$(;Y)
00317 F2=VAL(K1$): Y=SEARCH(K1$,"/")+1: K1$=K1$(;Y)
00318 F3=VAL(K1$): F4=100*F3+F2+F1/100: RETURN
00319 REM
00320 REM ------------- Print Screen Heading Sub. --------------
00321 REM
00322 CLS: PRINT K5$;: CURS 53,1: PRINT "by^L.R.Ford";
00323 CURS 16,1: INVERSE: PRINT "^*^Dreamcards^LedgerMaster^2^* "
00324 NORMAL: PRINT [A64^45];: RETURN
00325 REM
00326 REM ------------------ Print BASE Sub. -------------------
00327 REM
00328 GOSUB 322:PRINT"PAYMENT^CATEGORIES:^^(";B0$(0);")":K2$="Category"
00329 IF P=0 THEN RETURN  REM Payments
00330 FOR X=1 TO P: K0$=B0$(X): F1=B2(X): GOSUB 334: NEXT X: RETURN
00331 GOSUB 322: PRINT "RECEIPT^CATEGORIES:^^("; B0$(0); ")"
00332 IF R=0 THEN RETURN  REM Receipts
00333 FOR X=1 TO R: K0$=B1$(X): F1=B3(X): GOSUB 334: NEXT X: RETURN
00334 J=X+3: K=3: IF X>10 THEN LET J=J-10: K=35  REM Set cursor var's
00335 CURS K,J: PRINT CHR$(X+64); ".^"; K0$; SPC(13-LEN(K0$));
```

NOTES:

NOTES:

```
00336 IF F1>0 THEN PRINT [F10.2^F1]  REM Print any balance
00337 RETURN
00338 REM
00339 REM ---------------- "Counter" Print Sub. -----------------
00340 REM
00341 IF X>1 THEN 343 REM Don't re-print "WAIT" if Counter>0
00342 CURS 1,12: PRINT [A26^32] "*^WAIT^*" [A92^32]
00343 IF X>0 THEN CURS 28,13: PRINT [I4^X];  REM Print Counter
00344 CURS 0: RETURN
00345 REM
00346 REM ---------------- Save/Load Type Sub. ------------------
00347 REM
00348 K2$="Base^(1)": IF N>2 THEN LET K2$=K2$+",^Files^(2),^All^(3)"
00349 IF N>3 THEN LET K2$=K2$+",^Merge^(4)"
00350 GOSUB 246: H=INT(VAL(K1$)): IF H>N THEN LET H=0
00351 o=H: RETURN
00352 REM
00353 REM ------------- Initialise Arrays Sub. -----------------
00354 REM
00355 IF C=0 THEN 357 REM Print caution if existing ledger
00356 GOSUB 252: IF K1$="N" THEN RETURN REM Sure?
00357 CLEAR: K5$="": V=320: STRS(V*51+1400)  REM V is Max. No. of files
00358 DIM A0(V+2),A1(V+2),B0(20),B1(20),B2(20)
00359 DIM B3(20),B4(20),B5(20): GOSUB 361: GOSUB 362
00360 FOR X=C TO V+2: A0$(X)="": NEXT X: K1$="": RETURN
00361 FOR X=Y TO 20: B0$(X)="": B2(X)=0: NEXT X: RETURN
00362 FOR X=Y TO 20: B1$(X)="": B3(X)=0: NEXT X: RETURN
```



Pocket Programs

Commodore 64

MESSAGE ENCRYPTOR

While recently engrossed in some junk spy novel, I wondered if it were possible to write a simple BASIC message encryption/decryption program which was reasonably friendly and, at the same time, difficult to crack.

The result is attached. Although written on a Commodore 64, the program lines are straight BASIC, except for the printer commands, and shouldn't require many changes to be transportable to other machines.

The actual mechanics of the program are fairly simple:

1. The program is driven by a codeword which can be as long or as short as required. A codeword of five or more letters or numbers is required in the attached program. This can be amended with a change to lines 90 and 265.

2. It ascertains the ASCII value of each letter of the message input from the keyboard.

3. It finds the ASCII value of a selected letter from the codeword.

4. The program multiplies the two values together to produce a four-figure number, which is the code for each input letter.

5. The device used to take the code out of the realms of simple letter replacement is that the letter selected from the codeword as the multiplier changes one letter each cycle of the loop. This ensures that even if the same letter is entered twice, the output code value will be different.

6. A message can be encoded simply by using the codeword, and decoded by the same codeword, but it's almost impossible to crack any other way.

Peter Foye,
Balmain, NSW.

```
READY.
5 REM" **MESSAGE ENCRYPTOR**"
6 REM" **BY PETER FOYE"
7 REM
10 PRINT"■"
12 PRINTTAB(10)"MESSAGE ENCRYPTOR"
14 PRINTTAB(9)"━━━━━━━━━━━━━━━━━━━━━━━━"
20 PRINT"WHAT IS THE KEY FOR THIS MESSAGE?":INPUT K$
30 PRINT:PRINT:PRINT"TYPE YOUR MESSAGE"
35 OPEN 3,4
90 FOR X=1 TO 5
100 GET A$:IF A$=""THEN 100
130 B=ASC(A$)
140 C$=MID$(K$,X,1)
150 D=ASC(C$)
160 E=B*D
165 PRINT A$"..."E
167 PRINT#3,E
170 NEXT X
175 CLOSE 3
180 GOTO 90
190 REM
200 REM" **MESSAGE DECRYPTOR**"
220 REM" **BY PETER FOYE"
225 REM
230 PRINT"■"
240 PRINTTAB(10)"MESSAGE DECRYPTOR"
250 PRINTTAB(9)"━━━━━━━━━━━━━━━━━━━━━━━━"
260 PRINT"WHAT IS YOUR CODE FOR THIS MESSAGE?":INPUT K$
265 FOR X=1 TO 5
270 PRINT:PRINT:PRINT"TYPE THE NUMBERS":INPUT L
310 C$=MID$(K$,X,1)
320 D=ASC(C$)
330 B=L/D
335 E$=CHR$(B)
340 PRINT E$
342 OPEN 3,4
345 PRINT#3,E$
347 CLOSE 3
350 NEXT X
360 GOTO 265
READY
```

Microbee

GUESS3

The object of this game is to guess a three-letter word chosen at random by the computer. The game is over when you guess the word, or after 10 unsuccessful attempts. Guesses are checked by the computer and the following codes are displayed:

(Y) — A letter is in the correct position.
(N) — A letter is in the wrong position.
(-) — A letter is not correct.

The program's initial dictionary consists of 50 words, but many more could be added — the only restriction is the amount of memory available. Also note all words must contain three different letters. To enlarge the program's dictionary you simply add DATA statements containing the new words to the end of the program. You will also need to change the randomising number in line 210.

Lonnie Riley,
Banyo, QLD.

```
00100 REM ***** Guess a 3 letter word *****
00110 CLEAR : DIM W1(3), G1(3) : POKE 257,2
00120 CLS : LORES : C=0 : H=0 : T=0
00130 PLOT 0,0 TO 0,47 TO 64,47 TO 64,0 TO 0,0 : PLOT 47,0 TO 47,40
00140 CURS 72 : PRINT "G u e s s - 3" : PLOT 0,40 TO 64,40
00150 FOR X=257 TO 833 STEP 64 : CURS X : Y=X/64-3 : IF Y<10 THEN PRINT " ";Y;".
" ELSE PRINT Y"."
00160 NEXT X
00170 NORMAL
00180 CURS 38 : PRINT "(Y)  Correct Place" : CURS 102 : PRINT "(N)  Wrong Place"
 : CURS 166 : PRINT "(-)  No Match"
00190 REM ***** Computer chooses word *****
00200 RESTORE 590
00210 Y=INT(RND*50)+1
00220 FOR X=1 TO Y : READW1$ : NEXT X
00230 REM ***** Input a guess *****
00240 CURS 290 : PRINT "What is your guess :  "
00250 PLAY1 : CURS 354 : PRINT [A18 32]
00260 POKE 220,13 : CURS 354 : INPUT G1$
00270 REM ***** Check for 3 unique letters *****
00280 IF LEN(G1$) <> 3 THEN CURS 418 : PRINT "Please enter a 3 letter word!" : G
OTO 240
00290 IF G1$(;1,1) = G1$(;2,2) THEN 330
00300 IF G1$(;1,1) = G1$(;3,3) THEN 330
00310 IF G1$(;2,2) = G1$(;3,3) THEN 330
00320 GOTO 350
00330 CURS 418 : PRINT "No two letters are the same! " : GOTO 240
00340 REM ***** Compare guess with computers word *****
00350 C=0 : H=0 : T=T+1 : CURS 418 : PRINT [A29 32]
00360 FOR X=1 TO 3
00370 FOR Y=1 TO 3
00380 IF G1$(;X,X) <> W1$(;Y,Y) THEN 430
00390 IF X=Y THEN 420
00400 H=H+1
00410 GOTO 430
00420 C=C+1
00430 NEXT Y
00440 NEXT X
00450 X=201+T*64
00460 CURS 219+64*T : PRINT G1$
00470 IF C>0 THEN FOR Y=1 TO C : CURS X+(Y-1)*3 : PRINT "Y" : NEXT Y
00480 X=201+T*64+C*3
00490 IF H>0 THEN FOR Y=1 TO H : CURS X+(Y-1)*3 : PRINT "N" : NEXT Y
00500 X=201+T*64+(C+H)*3
00510 IF C+H<3 THEN FOR Y=1 TO (3-(C+H)) : CURS X+(Y-1)*3 : PRINT "-" : NEXT Y
00520 IF C=3 THEN CURS 610 : PLAY1;2;3;4 : PRINT "Congratulations!" : CURS 674 :
 PRINT "You guessed it!" : GOTO 550
00530 IF T=10 THEN CURS 610 : PRINT "Sorry!" : PLAY4,3;1,3 : CURS 674 : PRINT "T
he word was ";W1$ : GOTO 550
00540 GOTO 240
00550 CURS 930 : PRINT "Play again (Y/N)? "
00560 Z1$=KEY$ : IF Z1$="" THEN 560
00570 IF Z1$<>"Y" AND Z1$<>"N" THEN 550
00580 PRINT Z1$ : IF Z1$="Y" THEN 110 ELSE PRINT "Thanks for playing!" : END
00590 DATA "POT","PEG","PAT","PIN","PET","PEN","PUN","PUT","PUB","POD"
00600 DATA "TUG","TIN","TON","TOP","TEN","THE","THY","TIC","TOW","TRY"
00610 DATA "RUN","RIM","RIG","RAM","RUT","RAW","RIT","RED","RAP","RUB"
00620 DATA "SAT","SEA","SAW","SET","SAD","SUN","SUM","SIN","SON","SOD"
00630 DATA "BET","BEG","BED","BAD","BAN","BAT","BUT","BUN","BUG","BIT"
```

Sharp MZ-700

TIDY

Written for the Sharp MZ-700, using Disk BASIC, this utility prints out BASIC programs in an improved format.

Specifically, all FOR-NEXT loops are indented, REMarks are highlighted and one-line 'Breaks' are inserted before LABEL statements.

To be listed, the program must have been saved as a sequential file, using the 'A' option.

When run, 'Tidy' first prompts for a filename. Errors are not trapped, so if the file does not exist you will receive the standard error message and be returned to BASIC.

After the file has been opened, you will be asked to specify a format. First, you must nominate the screen or the printer as the output device. Entering 'P' (CR) will set the internal plotter to 80 columns and shorten the page length to 40 lines.

The next choice is between normal or extended spacing. Choosing extended spacing will insert the breaks before labels and around REMs.

The final question, whether to 'suppress' or not to suppress, controls whether or not the line number and REM statement are displayed.

Answering 'Y' will replace the line number and REM statement with the string '* * *', but will leave the rest of the line alone.

Hints on Conversion

In line 620, CHR$ (17) performs a linefeed on the Sharp's video display.

In line 630, CHR$(10) causes a linefeed on the printer.

Also in line 630, the asterisk between the logical expressions performs a logical 'AND'. On other systems, this line should probably read 'IF PD=2 AND T$="E" THEN'.

Line 560 opens a sequential data file whose name is in the string PN$, so data can be read from it on channel 1.

The LABEL statement is rather like the Beeb's DEF PROC. A name can be assigned to a line and GOTO and GOSUB commands can refer to the name, rather than the line number.

In line 130, Disk BASIC (nobody in their right mind would call it BASIC 5Z-008) treats EOF as an error trap of sorts. When the end of the file is reached, the program will execute the statements following THEN.

Matthew Cochrane,
Albany, WA.

```
10 REM         PRINT
20 REM LISTS BASIC PROGRAMS IN AN
30 REM IMPROVED FORMAT
40 REM PROGRAMS MUST HAVE BEEN SAVED
50 REM AS A SEQUENTIAL FILE
60 REM ie USING THE `A' OPTION
70 REM
80 REM BY M.COCHRANE APRIL 1985
90 REM
100 REM INITIALISE
110 GOSUB "INIT"
120 REM GET A LINE
130 IF EOF(#1) THEN CLOSE:END
140 INPUT#1,A$
150 IF A$="" THEN 120
160 Z$=""
170 REM GET LINE NO
180 FOR C=1 TO LEN(A$)
190 IF MID$(A$,C,1)=" " THEN Z$=SPACE$(6
-C)+LEFT$(A$,C):A$=RIGHT$(A$,LEN(A$)-C):
GOTO 210
200 NEXT C
210 REM A LABEL
220 IF LEFT$(A$,5)="LABEL" THEN Z$=CS$+Z
$
230 REM A REM
240 IF LEFT$(A$,3)="REM" THEN GOSUB "REM
ARK":GOSUB "PRINT":GOTO 120
250 REM LOOK FOR `FOR'
260 S$="FOR "
270 GOSUB "SEARCH"
280 NI=NI+NO
290 REM LOOK FOR `NEXT'
300 S$="NEXT "
310 GOSUB "SEARCH"
320 NI=NI-NO
330 IF NI<0 THEN NI=0
340 REM FINAL FORMAT
350 Z$=Z$+SPACE$(NI)+A$
360 IF PLR=-1 THEN PLR=0:Z$=CS$+Z$
```

Sharp MZ-700

```
370 GOSUB "PRINT"
380 GOTO 120
390 LABEL "PRINT"
400 ON PD GOTO 410,420
410 PRINTZ$:RETURN
420 PRINT/P Z$:RETURN
430 LABEL "SEARCH"
440 NO=0
450 QF=0
460 FOR C=1 TO LEN(A$)-LEN(S$)
470 IF MID$(A$,C,1)=CHR$($22) THEN QF=1-
QF
480 IF QF=1 THEN 500
490 IF MID$(A$,C,LEN(S$))=S$ THEN NO=NO+
500 NEXT C
510 RETURN
520 LABEL "INIT"
530 NI=0
540 PLR=0
550 INPUT "File Name       ?";PN$
560 ROPEN#1,"QD:"+PN$
570 INPUT "Output    (P/S) ?";T$
580 PD=1
590 IF T$="P" THEN PD=2:MODE TS:PAGE 40
600 INPUT "Spacing   (N/E) ?";T$

610 CS$=""
620 IF T$="E" THEN CS$=CHR$(17)
630 IF (PD=2)*(T$="E") THEN CS$=CHR$(10)
640 INPUT "Suppress (Y/N) ?";T$
650 RS=0
660 IF T$="Y" THEN RS=-1
670 RETURN
680 LABEL "REMARK"
690 IF RS THEN Z$="* * *":A$=RIGHT$(A$,L
EN(A$)-3)
700 Z$=Z$+A$
710 IF PLR=0 THEN Z$=CS$+Z$:PLR=-1
720 RETURN
```

Commodore 64

WILDERNESS

This fantasy game was written by my brother Paul and myself, and is similar to 'Dungeons and Dragons', in that you wander around searching for treasure and fighting monsters. It includes a dungeon, which may be entered from the wilderness, and as you venture forth, fighting monsters (greblies) along the way, your character gains experience points, which allow you to go up in levels of skill. The game is complete and runs bug-free (as far as we can tell), but as yet has no defined ending. Due to a lack of programming utilities (renumber and so on), it became too difficult for us to go on expanding the game, but features we intended to include were:

- Magic spells
- Scrolls
- Amulets
- Friendly/unfriendly encounters
- Magic books
- Assorted treasures
- More monsters
- Crynts
- Villages
- 'Save game' feature
- Night/day
- More weapons
- Divine intervention (the gods).

Perhaps readers will be able to incorporate these features themselves.

The commands are:

m — (move) advances you to the next screen in the wilderness or next room in the dungeon.
s — (search) searches the area/room for treasure/weapons.
x — (status) displays your character's status, strength, possessions and hit points.
c — (print commands) displays the available commands.
a — (attack) attacks the monster in your area/room. The monster must be in range to be attacked; the 'move' command advances you into range.
i — (identify) identifies what kind of monster is on screen/in the room.

```
LISTING OF 'WILDERNESS'
FOR THE COMMODORE 64
25/9/85

WRITTEN BY PAUL STUART
WITH MODIFICATIONS
BY ANDREW STUART

ORIGINAL CONCEPT FROM
THE GAME 'WILDERNESS'
FOR THE SORCEROR COMPUTER.
CONVERTED AND RE-WRITTEN
FOR THE COMMODORE 64 BY
PAUL STUART

PLEASE FEEL FREE TO MODIFY
OR TO CONVERT TO OTHER
SYSTEMS.


10 GOSUB 7300
200 REM ********************************
210 REM *                              *
220 REM *        DUNGEON               *
230 REM *                              *
240 REM ********************************
250 REM   WRITTEN BY PAUL STUART
260 REM        WITH   ANDREW STUART
270 REM
280 REM       DATE:25/4/84
290 REM
300 REM       VERSION 7
310 REM
320 REM
330 REM
340 REM*******SPRITE CREATION*********
350 REM
430 POKE 53280,0 : POKE 53281,0
432 POKE 646,11
440 PRINT CHR$(147)
442 V=53248
451 POKE V+21,4 : REM ENABLES SPRITE 2
453 POKE 2042 ,13 : REM DATA LOCATION
455 FOR N=0 TO 15:POKE 832+N,0:NEXT N
457 FOR N=0 TO 30:READ Q:POKE 848+N,Q:      NEXT N
459 FOR N=0 TO 15:POKE 879+N,0:NEXT N
461 DATA 48,0,0,72,0,0,72,0,0,48,0,0
462 DATA 252,0,0,48,0,0,48,0,0,48
463 DATA 0,0,72,0,0,72,0,0,204
730 REM
740 DIM W$(12)
745 DIM DUNJ(22,5)
750 GOTO 1200
755 REM
760 REM ****** DUNGEON GENERATION ******
765 REM
766 REM DUNJ(RM YOU ARE IN,0) GIVES ROW
767 REM NO OF DUNJ WHICH HAS DETAILS OF
768 REM THAT ROOM
770 RESTORE
775 FOR M=1 TO 31 :READ Q:NEXT M
780 FOR D=1 TO 22
800 R=INT(RND(1)*22)+1
820 FOR E=1 TO D
840 IF DUNJ(E,1)=R THEN 800
860 NEXT E
880 DUNJ(D,1)=R :DUNJ(R,0)=D:          DUNJ(D,5)=INT(RND(1)*10)+1
900 NEXT D
905 CN=INT(RND(1)*22)+1
930 FOR L =1 TO 22
940 READ L1,L2,L3
960 DUNJ(L,2)=DUNJ(L1,1)
```

Commodore 64

r — (run) is for when you're about to lose a fight.
se — (search) is used to enter a dungeon and find out if there's another way out.
l — (look) redraws the wilderness map or describes the current dungeon chamber again.

Paul has attempted to write the program with some sort of structure; it's composed of a number of separate subroutines to make expansion or conversion a little easier. We've also attempted to use as few of those funny-looking Commodore graphics characters as possible; where they do appear they're only for screen formatting.

We would like feedback on the program; if there are any enquiries, bugs found, suggestions (no criticism, please), or if anyone wants a disk or tape version, please contact Paul on (03) 29 1949 or me (Andrew) on (03) 836 0775.

Andrew Stuart,
Surrey Hills, VIC.

```
980 DUNJ(L,3)=DUNJ(L2,1)
1000 DUNJ(L,4)=DUNJ(L3,1)
1020 NEXT L
1040 DATA 2,3,5,1,3,5,1,2,4,3,5,6,1,2,4
1060 DATA 4,7,9,6,8,11,7,9,10,6,8,14,8
1080 DATA 12,13,7,15,16,10,13,17,10,12,17
1100 DATA 9,18,19,11,16,22,11
1120 DATA 15,22,12,13,21,20,19,14,20,18
1140 DATA 14,18,19,21,20,17,22,15,16,21
1170 TR$="NOTHING OF VALUE"
1175 RETURN
1180 REM
1200 REM *******CHARACTER CREATION*****
1220 REM
1240 PM=30 : HP=30  : DO=0
1260 SB=INT(RND(1)*7)-1
1280 IF SB<0 THEN SB=0
1300 DB=INT(RND(1)*6)-1
1320 IF DB<0 THEN DB=0
1340 W$(1)="1.  BARE HANDS"
1360 W$(2)="2.  DAGGER"
1380 W$(3)="3.  SWORD"
1390 ARMOUR$(1)="1. LEATHER ARMOUR":AA=1
1400 LEVEL=1:XP=0 : CD=20 : CA=20 : W=3      :WF=30
1405 TR$="NOTHING OF VALUE"
1410 PRINT CHR$(147)
1415 POKEV+4,0 : POKE V+5,0
1420 GOSUB 5510
1460 PRINT CHR$(147)
1480 GOSUB 5710
1500 PRINT CHR$(147)
1510 GOTO 1930
1520 REM
1540 REM **********MOVEMENT*********
1560 REM
1570 PRINT
1580 PRINT "TO WHICH ROOM:";B;C;D;
1600 INPUT A%
1610 IF A%=B THEN FF=B: GOTO 1650
1620 IF A%=C THEN FF=C : GOTO 1650
1630 IF A%=D THEN FF=D : GOTO 1650
1640 PRINT " HOW ?" : GOTO 1580
1650 CN=DUNJ(FF,0)
1660 IF HP<PM THEN HP=HP+3
1670 TR$="NOTHING OF VALUE"
1760 REM
1770 REM*********POSITION*************
1780 REM
1781 POKE V+5,0:POKE V+4,0
1782 PRINT CHR$(147)
1783 PRINT: PRINT
1785 A=0:B=0:C=0:D=0
```

NOTES:

Commodore 64

```
1790 B=DUNJ(CN,2) : C=DUNJ(CN,3)
1800 D=DUNJ(CN,4) : A=DUNJ(CN,1)
1810 PRINT "YOU ARE IN ROOM ";A
1815 PRINT
1820 PRINT "THIS CHAMBER JOINS ";B",";C",";D :RETURN
1842 REM
1845 REM********WILDERNESS************
1846 REM
1847 IF DO=1 THEN 1950
1848 PRINT CHR$(147)
1849 POKE V+4,0:POKE V+5,0
1850 PRINT "┌──────────────┐"
1855 FOR M=1104 TO 1544 STEP 40
1857 POKE M+54272,11 :REM POKE COLOUR
1860 POKE M,71
1865 E=INT(RND(1)*5)+1
1870 FOR G=1 TO E
1872 H=INT(RND(1)*16)+1
1873 IF M=1344 AND H=9 THEN 1872
1875 IF M=1384 AND H=9 THEN 1872
1877 POKE M+H+54272,5
1880 POKE M+H,65
1885 NEXT G
1887 POKE M+17+54272,11
1890 POKE M+17,66
1895 NEXT M
1900 PRINT "■■■■■■■■■■■■└──────────────┘"
1905 POKE V+4,90
1910 POKE V+5,110
1911 IF HP<PM THEN HP=HP+1
1912 DUNJ=INT(RND(1)*3)+1
1913 IF DUNJ<>1 THEN RETURN
1914 X1=INT(RND(1)*14)+1
1915 IF HP<PM THEN HP=HP+1
1916 Y1=INT(RND(1)*10)+4
1917 POKE X1+40*Y1+1024+54272,6
1918 POKE X1+40*Y1+1024,4
1919 RETURN
1920 RETURN
1930 REM*******************************
1935 IF DO=1 THEN GOSUB 1540
1937 IF DO=0 THEN GOSUB 1845 :REM WILD
1940 GOTO 2200    :REM GREBLIE CHECK
1950 GOSUB 4610   :REM TEASURE CHECK
1955 REM
1960 REM********COMMANDS*************
1970 REM
1980 PRINT : PRINT
1995 A$=""
2000 INPUT "YOUR MOVE";A$
2010 IF DO=0 THEN GOSUB 7000
2020 IF A$="M" THEN 1930
2030 IF A$="X" THEN GOSUB 5510:GOTO 1960
2040 IF A$="S" THEN GOSUB 5910:GOTO 1960
2045 IF A$="SE" AND DO=1 THEN GOSUB 2150     :GOTO 1960
2047 IF A$="SE" AND DO=0 THEN GOTO 6090
2050 IF A$="C" THEN GOSUB 5710:GOTO 1960
2055 IF A$="L" AND DO=1 THEN GOSUB 1770:     GOTO 1960
2057 IF A$="L" AND DO=0 THEN GOSUB 1845:      GOTO 1960
2060 IF A$="" THEN GOTO 1960
2070 IF A$="A" THEN PRINT:PRINT"YOU ARE ALONE" : GOTO 1960
2080 IF A$="I" THEN PRINT "NO VISIBLE ENEMY" : GOTO 1960
2085 IF A$="Z" THEN END
2090 PRINT "I DONT UNDERSTAND.":GOTO 1960
2150 REM*********EXIT***************
2151 PRINT:PRINT "SEARCHING......."
2152 FOR M=1 TO 500:NEXT M
2155 M=INT(RND(1)*5)+1
2160 IF M<>1 THEN PRINT :PRINT              "YOU CAN FIND NO EXIT HERE!":RETURN
2165 PRINT
2170 INPUT"EXIT FOUND,GOING UP";A$
2172 IF A$<>"Y" THEN RETURN
2175 IF A$="Y" THEN DO=0 :PRINT:PRINT       "ENTERING WILDERNESS"
```

Commodore 64

```
2177 FOR M=1 TO 750:NEXT M:PRINT CHR$(147)
2180 PRINT "ENTRANCE SEALS BEHIND YOU"        :FOR M=1 TO 750:NEXT M
2185 GOSUB 1845
2190 RETURN
2200 REM*****RANDOM GREBLIE**********
2205 REM
2210 Q=INT(RND(1)*3)+1
2220 IF Q=1 THEN 2230
2225 IF DO=1 THEN GOTO 1845
2227 GOTO 1950 :REM WILDERNESS MODE
2229 REM
2230 REM******GREBLIE CREATION*******
2235 REM
2240 M= INT (RND(9)*13)+1
2250 ON M GOTO 2260,2270,2280,2290,2300,2310,2320,2330,2340,2350,2360,2370,2380
2260 M$="GERG":MA=25:MD=55:NA=6:DM=4          :MHP=6:AD=1:GOTO 2460
2270 M$="LERG":MA=30:MD=60:NA=6:DM=6          :MHP=4:AD=4:GOTO 2460
2280 M$="GOBLIN":MA=40:MD=60:NA=4:DM=8:       MHP=7:AD=0:GOTO 2460
2290 M$="HUMAN":MA=35:MD=65:NA=4:DM=6:         MHP=4:AD=2:GOTO 2460
2300 M$="CORGURNG":MA=30:MD=70:NA=4:          DM=4:MHP=10:AD=4:GOTO 2460
2310 M$="TROGLODYTE" :MA=30:MD=80:NA=6:       DM=5:MHP=5:AD=5:GOTO 2460
2320 M$="THROON":MA=35:MD=70:NA=6:DM=4:        MHP=6 :AD=5 :GOTO 2460
2330 M$="OGRE":MA=40:MD=75:NA=1:DM=6:         MHP=12:AD=4:GOTO 2460
2340 M$="WHELK ":MA=47:MD=90:NA=4:DM=6:       MHP=8:AD=5:GOTO 2460
2350 M$="ORC":MA=45:MD=80:NA=2:DM=8:          MHP=7:AD=5:GOTO 2460
2360 M$="BIG WORM":MA=35:MD=65:NA=1:          DM=4:MHP=10:AD=6:GOTO 2460
2370 M$="WARG":MA=45:MD=60:NA=5:DM=5:         MHP=4:AD=1:GOTO 2460
2380 M$="NAGRUTHA":MA=30:MD=70:NA=1:          DM=6:MHP=6:AD=3:GOTO 2460
2460 REM******GREBLIE DETAILS**********
2470 REM
2480 NUMBER=INT(RND(1)*NA)+1+(LEVEL-1)
2490 MP=INT(RND(1)*MHP)+1+AD
2500 EX=(MA+MD+DM++MP)*NUMBER
2510 GOSUB 4610
2515 IF DO=1 THEN GOTO 2540
2520 Y=INT(RND(1)*10)+4
2522 X=INT(RND(1)*15)+2
2526 GOTO 2689
2528 GOTO 2550
2530 PRINT
2535 PRINT NUMBER;M$;" BLOCK YOUR WAY"
2540 PRINT
2545 PRINT NUMBER;M$;" BLOCK YOUR WAY"
2550 REM******MELEE COMMAND***********
2560 REM
2570 PRINT
2580 PRINT " YOUR MOVE ";
2585 A$=""
2590 INPUT A$
2595 IF DO=0 THEN GOSUB 7000
2600 IF A$="M" AND DO=1 THEN PRINT:PRINT:"THEY WONT LET YOU PASS":GOTO 2550
2610 IF A$="M" AND DO=0 THEN 2655
2620 IF A$="X" THEN GOSUB 5510:GOTO 2550
2625 IF A$="A" THEN 2761
2630 IF A$="S"THEN GOSUB 5910:IF (X>=7 AND X<=11)AND(Y>=7 ANDY<=11) THEN 3510
2631 IF A$="S" THEN 2550
2632 IF A$="L" AND DO=1 THEN GOSUB 1770:      GOTO 2550
2633 IF A$="L" THEN GOTO 2655:GOTO 2550
2634 IF A$="SE" AND DO=0 THEN 6090:           GOTO 2550
2635 IF A$="SE" AND DO=1 THEN GOSUB 2150      :IF DO=0 THEN 1960
2636 IF A$="SE" AND DO=1 THEN GOTO3510
2640 IF A$="R" THEN 2700
2645 IF A$="I" THEN PRINT "GREBLIES ARE";NUMBER;M$ :GOTO 2550
2647 IF A$="Z" THEN END
2649 IF A$="" THEN 2550
2650 PRINT "I DONT UNDERSTAND THAT COMMAND":GOTO 2550
2654 REM
2655 REM********GREBLIE MOVER*********
2657 REM
2660 IF(7<=X AND X<=11)AND (Y<=11AND Y>=7) THEN 3510
2665 E=INT(RND(1)*5)+1
2670 IF X<7 THEN X=X+E
2672 IF X>11 THEN X=X-E
```

Commodore 64

```
2675 E=INT(RND(1)*5)+1
2680 IF Y<7 THEN Y=Y+E
2685 IF Y>11 THEN Y=Y-E
2687 GOSUB 1845 :REM WILDERNESS
2689 IF (7<=X AND X<=11)AND (Y<=11AND Y>=7) THEN PRINT "GREBLIES ARE IN RANGE"
2690 POKE X+40*Y+1024+54272,11
2693 POKE X+40*Y+1024,13
2695 GOTO 2550 : REM MELEE COMMAND
2700 REM
2710 REM********RETREATING***********
2715 REM
2720 M=INT(RND(1)*10)+1
2730 IF M<5 AND DO=1 THEN FF=D:PRINT          "YOU MADE IT":GOSUB 1770:GOTO 1960
2735 IF M<5 AND DO=0 THEN PRINT               "YOU MADE IT"
2737 FOR Y = 1 TO 400:NEXT Y :GOTO 1930
2740 PRINT "YOU DIDNT MAKE IT"
2750 FOR Y= 1 TO 400:NEXT Y
2760 REM
2761 REM*******WEAPON CHOOSING*********
2762 REM
2780 REM
2782 IF DO=1 'THEN GOTO 2790
2785 IF (7=<YANDY=<11) AND (7=<XANDX=<11) THEN 2790
2786 PRINT "GREBLIES ARE NOT IN RANGE":       GOTO 2550
2790 DAMAGE=0:KILLED=0:HIT=0:R=0
2800 FOR M=1 TO W
2810 PRINT W$(M)
2820 NEXT M
2825 PRINT "WHICH WEAPON";
2830 INPUT A%
2835 M=0
2840 M=M+1
2850 IF VAL(W$(M))=A% THEN 2900
2855 IF M=12 THEN 2870
2860 GOTO 2840
2870 PRINT "YOU DON'T HAVE THAT WEAPON"
2880 GOTO 2830
2900 ON A% GOTO 2910,2920,2930,2940,2950,2960,2970,2980,2981,2982,2983,2984
2910 WA=15:WD=2: GOTO 3100
2920 WA=20: WD=4 : GOTO 3100
2930 WA=30: WD=6: GOTO 3100
2940 WA=30: WD=8:GOTO 3100
2950 WA=35: WD=10 :GOTO 3100
2960 WA=25: WD=12 :GOTO 3100
2970 WA=35 :WD=10 :GOTO 3100
2980 XXXXXXRUNESTAFFXXXXXX
2981 WA=40 : WD=10 : GOTO 3100
2982 WA=30 : WD=12 : GOTO 3100
2983 REM XXXXXXXXARROWXXXXXXXX
2984 WA=45 : WD=8 : GOTO 3100
2985 WA=50 : WD=6 : GOTO 3100
2986 XXXXXXX
2987 XXXXXXXXXX
2988 WA=40 : WD=15 : GOTO 3100
2989 XXXXXHORN
2990 WA=50 : WD=12 : GOTO 3100
2991 WA=60 : WD=15 : GOTO 3100
2992 XXXXXX
3100 REM********PLAYER ATTACK***********
3105 REM
3107 IF DO=0 THEN GOSUB 7000
3110 HIT=(CA+WA+SB+LEVEL)/MD
3115 PRINT "CHANCE TO HIT:";INT(HIT*100);"%"
3120 R=RND(1):PRINT "RESULT:";INT(R*100);"%"
3140 PRINT
3150 PRINT "YOUR ATTACK ";
3155 FOR M=1 TO 700  :NEXT M
3160 IF R<HIT THEN 3170
3162 PRINT "MISSES"
3165 FOR M=1 TO 1000 : NEXT M
3167 GOTO 3510
3170 DAMAGE=INT(RND(1)*WD)+1+LEVEL+SB
3175 PRINT "HITS" ,"DAMAGE:";DAMAGE
```

Commodore 64

```
3180 DAMAGE=DAMAGE+RMAIN
3190 IF DAMAGE=0 THEN 3230
3200 PRINT
3205 IF DAMAGE>MP THEN KILLED=KILLED+1:       IF DAM>MP THEN DAM=DAM-MP:GOTO3205
3210 NUMBER=NUMBER-KILLED
3220 RMAIN=DAMAGE
3230 IF NUMBER<1 THEN 3280
3240 PRINT
3250 PRINT "GREBLIES SLAIN:";      KILLED
3255 FOR M=1 TO 1000 :NEXT M
3260 KILLED=0 : GOTO 3510
3270 REM ALL GREBLIES KILLED
3280 KILLED=KILLED+NUMBER
3290 PRINT
3300 PRINT "GREBLIES SLAIN:";      KILLED
3310 XP=XP+EX
3320 IF XP>(LEVEL*1000) THEN 3360
3330 RMAIN=0 : NUMBER=0 : DAMAGE=0
3340 KILLED=0:Q=0
3345 FOR M=1 TO 1000 : NEXT M
3347 IF DO=0 THEN GOSUB 7000
3350 GOTO 1960
3360 LEVEL=LEVEL+1
3370 M=INT(RND(1)*5)+1
3380 PM=PM+M
3385 PRINT
3390 PRINT "YOU HAVE REACHED LEVEL";LEV
```

Commodore 64

```
3395 FOR M=1 TO 2000 : NEXT M
3400 GOTO 3330
3500 REM
3510 REM*********GREBLIE ATTACK********
3520 REM
3525 IF DO=0 THEN GOSUB 7000
3530 HIT=MA/(CD+WF+LEVEL+DB)
3535 PRINT "CHANCE TO HIT:";INT(HIT*100);"%"
3540 PRINT
3550 M=1
3560 PRINT M$;" ATTACKS ";
3570 R=RND(1)
3580 FOR E=1 TO 1000 : NEXT E
3590 IF R<HIT THEN 3640
3600 PRINT "MISS"
3605 IF M=NUMBER THEN 3620
3610 M=M+1 : GOTO 3560
3620 FOR M=1 TO 1000 : NEXT M
3622 IF DO=0 THEN GOSUB 7000
3625 GOTO 2550
3627 REM
3640 DAMAGE=INT(RND(1)*DM)+1
3660 PRINT "HIT       DAMAGE:";DAMAGE
3670 HP=HP-DAMAGE
3675 IF HP<1 THEN 3910
3680 GOTO 3605
3900 REM
3910 REM*******DEATH SECTION**********
3920 REM
3922 RMAIN=0:Q=0:DAMAGE=0
3925 PRINT:PRINT :PRINT
3930 PRINT CHR$(147)
3935 POKE V+4,0:POKE V+5,0
3940 PRINT "THE CHARACTER DIED ......";
3950 PRINT "TOUGH."
3960 PRINT "YOU REACHED LEVEL";LEVEL;          "BEFORE YOU"
3965 PRINT"WERE GROTTED."
3970 PRINT
3990 INPUT "TRY AGAIN (Y/N)";A$
4000 IF A$="Y" THEN 1200
4010 END
4020 REM
4220 PRINT "A STAIRWAY OPENS UP BEFORE";3980
4490 REM
4500 REM*******PAUSE ROUTINE**********
4510 REM
4515 PRINT
4520 PRINT"PRESS ANY KEY TO CONTINUE";
4522 GET A$:IF A$="" THEN 4522
4530 RETURN
4600 REM
4610 REM********TREASURE CHECK*********
4620 REM
4621 IF DO=0 THEN 4630
4622 IF DUNJ(CN,5)=1 THEN DUNJ(CN,5)=0:        GOTO 4670
4625 RETURN
4630 M=INT(RND(1)*4)+1
4640 IF M=1 THEN 4670
4650 RETURN
4660 REM
4670 REM ******TREASURE DETERMINER*****
4680 REM
4700 M=INT(RND(1)*(LEVEL+9))+1
4710 ON M GOTO 4720,4730,4740,4750,4750,  4770,4780,4790,4800,4810,4810,4810
4720 TR$="4. MORNING STAR": RETURN
4730 TR$="5. BATTLE AXE": RETURN
4740 TR$="6. SILVER DAGGER" : RETURN
4750 TR$="7. LLOYD SWORD": RETURN
4760 TR$="8. RUNESTAFF" : RETURN
4770 TR$="9. TROLLS ARM": RETURN
4780 TR$="10. SERPENT STAFF":RETURN
4790 TR$="11. CRYSTAL ARROW":RETURN
4800 TR$="12. MACE OF DISRUPTION":RETURN
```

Commodore 64

```
4810 TR$="13. FIRE HAMMER":RETURN
4820 TR$="14. ":RETURN
4830 TR$="15. ":RETURN
4840 TR$="16. VORPAL BLADE":RETURN
4850 TR$="17. HORN OF SUMMONING":RETURN
4860 TR$="18. TRIDENT OF DAEMONS":RETURN
4870 TR$="19. STORMBRINGER":RETURN
4880 TR$="20. ":RETURN
5500 REM
5510 REM******DISPLAY STATUS************
5520 REM
5530 PRINT
5540 PRINT "HIT POINTS=";HP
5550 PRINT "STRENGTH BONUS=";SB
5560 PRINT"DEXTERITY BONUS=";DB
5570 PRINT "YOUR WEAPONS ARE:"
5580 FOR M=1 TO W
5590 PRINT TAB(17); W$(M)
5600 NEXT M
5605 PRINT "YOU ARE WEARING";ARMOUR$(AA)
5610 PRINT "YOUR LEVEL=";LEVEL
5620 PRINT"EXPERIENCE POINTS=";XP
5630 GOSUB 4500
5640 RETURN
5700 REM
5710 REM*******DISPLAY COMMANDS********
5720 REM
5740 PRINT TAB(2);"COMMANDS:":PRINT
5750 PRINT TAB(4);"M MOVE",
5760 PRINT "S SEARCH"
5770 PRINT TAB(4);"X STATUS",
5780 PRINT "C PRINT COMMANDS"
5790 PRINT TAB(4);"A ATTACK",
5800 PRINT "I IDENTIFY GREBLIE"
5810 PRINT TAB(4);"R RUN    ",
5820 PRINT "SE SEARCH ENTRANCE"
5825 PRINT TAB(4);"SE EXIT",
5830 PRINT "L LOOK"
5840 GOSUB 4500
5860 IF DO=0 THEN GOSUB 7000
5880 RETURN
5900 REM
5910 REM*********SEARCHING SECTION*****
5920 REM
5924 FOR M=1 TO W
5925 IF W$(M)=TR$ THEN TR$="NOTHING OF VALUE"
5926 NEXT M
5930 PRINT
5940 PRINT "YOU FIND ........";
5950 FOR M=1 TO 500: NEXT M
5960 PRINT TR$
5970 IF TR$="NOTHING OF VALUE" THEN RETURN
5980 W=W+1
5990 W$(W)=TR$
6000 TR$="NOTHING OF VALUE"
6010 RETURN
6080 REM
6090 REM*********ENTRANCE ROUTINE******
6095 REM
6096 A$="NO ENTRANCE TO THE UNDERWORLD CAN BE FOUND"
6097 IF DUNJ<>1 THEN PRINT A$:GOTO 6099
6098 GOTO 6105
6099 IF Q=1 THEN GOTO 2550
6100 IF Q<>1 THEN 1960
6105 PRINT "ENTRANCE TO DUNGEON FOUND,";
6110 A$=""
6120 INPUT "GOING DOWN";A$
6140 IF A$<>"Y" THEN 6099
6150 POKE V+4,0:POKE V+5,0
6155 DUNJ=0
6157 PRINT CHR$(147)
6158 FOR Y = 1 TO 11:PRINT:NEXT Y
6160 PRINTTAB(6)"DESCENDING INTO UNDERWORLD."
6170 GOSUB 760   :REM DUNJ GENERATER
6180 DO=1 :Q=0:  GOSUB 1770
6190 GOTO 1960
7000 REM
7020 REM *******SCREEN CLEAR ********
7040 REM
7060 PRINT CHR$(19)
7080 FOR M=1 TO 7
7100 PRINT CHR$(17)  :REM CURSOR DOWN
7120 NEXTM
7140 FOR M=1 TO 9
7160 PRINT "
7180 NEXT M
7200 PRINT "[REVERSE]         [OFF]"
7260 RETURN
7300 REM *********INTRO SCREEN********
7301 PRINT CHR$(147)
7302 POKE 53272,22
7303 POKE 53280,0:POKE 53281,0
7304 POKE 646,11
7305 V=53248
7306 POKE V+5,0
7307 POKE V+4,0
7310 FOR Y=1 TO 4
7320 PRINT CHR$(17)
7330 NEXT Y
7340 PRINT TAB(14)"oILDERNESS!"
7350 PRINT
7360 PRINT TAB(11)"♠ GAME OF FANTASY."
7370 FOR Y= 1 TO 3000:NEXT Y
7380 PRINTCHR$(147):POKE 53272,21:RETURN

READY.
```

TRS-80 MC10

TOWER OF HANOI

The object of the game is to move the stack of counters from left to right, while abiding by two rules:

- Move only one counter at a time.
- A larger counter cannot sit on top of a smaller one.

There are two positions to the right of the original stack. My program does not permit you to hop an intervening space, as would be possible (but not permissible) in the original.

J. L. *Elkhorne*,
East Malvern, VIC.

```
1 REM tower of hanoi
2 REM 13/6/84 - j.l. elkhorne
3 DIM A(4,2):DIM B(4,2)
4 :
50 B(0,0)=320:B(1,0)=352:B(2,0)=384:B(3,0)=416:B(4,0)=448
55 B(0,1)=331:B(1,1)=363:B(2,1)=395:B(3,1)=427:B(4,1)=459
60 B(0,2)=342:B(1,2)=374:B(2,2)=406:B(3,2)=438:B(4,2)=470
90 :
100 REM patterns-colours 2-6
110 S$(1)="         ":REM 9 BLKS
120 S$(2)="        ":REM 1 YEL
130 S$(3)="   ///   ":REM 3 BLU
140 S$(4)="  ?????  ":REM 5 RED
150 S$(5)=" OOOOOOO ":REM 7 BUF
160 S$(6)="_________":REM 9 CYAN
170 CLS:GOTO 1000
190 :
200 REM search (point is v,h)
210 R=0:C=0
230 FOR H=20 TO 28 STEP 2
240 :  FOR V= 8 TO 52 STEP 22
250 A(R,C)=POINT(V,H)
260 C=C+1:NEXT
270 C=0:R=R+1:NEXT
280 RETURN
290 :
```

TRS-80 MC10

```
300 REM prompt for move
310 PRINT@ 32,"1.":REM YELLOW SQ.
320 PRINT@ 64,"2./":REM BLUE
330 PRINT@ 96,"3.?":REM RED
340 PRINT@128,"4.O":REM BUFF
350 PRINT@160,"5. ":REM CYAN
360 PRINT@224,"ENTER COLOUR NUMBER AND","DIRECTION (L/R) - AS 1R"
370 PRINT@288:RETURN
390 :
400 REM set-up
410 PRINT@320,S$(2):PRINT@352,S$(3)
420 PRINT@384,S$(4):PRINT@416,S$(5)
430 PRINT@448,S$(6):RETURN
490 :
500 REM move
510 PRINT@288,:INPUT P$
520 B=VAL(P$):D$=MID$(P$,2,1)
540 IF B<1 OR B>5 OR D$<>"R" AND D$<>"L" THEN PRINT@288:GOTO 510
550 B=B+1:RETURN
590 :
1000 REM mainline
1010 CLS:GOSUB 400
1020 GOSUB 300:GOSUB 500:T=T+1: GOSUB 200
1090 :
1100 REM locate
1110 FOR R=0 TO 4:FOR C=0 TO 2
1120 IF A(R,C)=B THEN 1200
1130 NEXT:NEXT
1190 :
1200 REM move test
1210 IF D$="L" AND C=0 THEN PRINT@70,"WRONG DIRECTION!":SOUND24,9:GOTO 1020
1220 IF D$="R" AND C=2 THEN PRINT@70,"WRONG DIRECTION!":SOUND 42,8:GOTO 1020
1230 IF R=0 THEN 1300
1240 IF A(R-1,C)<>-1 THEN PRINT@70,"NOT THE TOP!":SOUND110,13:GOTO 1020
1290 :
1300 REM move
1310 R0=R:C0=C:R=0
1320 IF D$="L" THEN C=C-1
1330 IF D$="R" THEN C=C+1
1350 IF R=4 THEN 1400
1360 IF A(R,C)=-1 THEN R=R+1
1370 IF A(R,C)>B THEN R=R-1:GOTO 1400
1375 IF A(R,C)>-1 AND A(R,C)<B THEN PRINT@70,"SMALLER UNDER!":SOUND 44,14:GOTO
1380 GOTO 1350
1390 :
1400 REM new
1410 PRINT@B(R,C),S$(B);
1415 PRINT@B(R0,C0),S$(1);
1420 R=5:C=3
1490 :
1500 REM test end?
1510 IF POINT(52,20)=2 THEN 1700
1520 GOTO 1020
1590 :
1700 REM score
1710 PRINT@0,"YOU TOOK";T;"TURNS."
1720 FOR R=1 TO 8:SOUND RND(144),4:NEXT
1730 GOTO 1730
1999 END
```

Microbee

GOLDEN EAGLE POKIES

This game will cost you a dollar for every pull of the handle, and you can win some cash by spinning one of the winning combinations. The characters rank in order of value from the 'gold bar' to the 'lemon' (this is explained in the program's instructions). When the game is running, you simply enter Y or N to indicate whether you want to have another spin or exit the program. Note: pressing return gives the same result as entering Y.

James Saunders,
Riverside, TAS,

```
00100 REM ##########################
00110 REM # By J.Saunders 6-10-85 #
00120 REM ##########################
00130 DIM Z1(6)
00140 V=5:M=0:B=11
00150 CLS:NORMAL:CURS 10,1:PRINT "*** WE
LCOME TO THE GOLDEN EAGLE POKIES! ***"

00160 CURS 2,4:PRINT"This is a Poker Gam
e.  It will cost you one dollar for each
"\"spin.  You will be given five dollars
 credit to begin with,"
00170 PRINT"and you may win more money b
y spinning up one of the winning"\"combi
nations.  These combinations are listed
on the next page."
00180 PRINT"The game is quite simple and
 self explanatory, so Goodluck!"\
00190 DATA 0,0,31,63,127,255,255,255,255
,255,255,255,127,63,31,15
00200 DATA 227,118,12,152,153,255,255,25
5,255,255,255,255,255,255,195,0
00210 DATA 128,0,248,252,198,143,31,255,
255,255,255,255,254,252,248,240
00220 DATA 1,1,0,0,0,0,0,0,0,0,1,3,7,7,3,1
,0
00230 DATA 195,230,236,24,24,126,255,255
,255,249,241,243,255,255,255,255
00240 DATA 0,0,0,0,0,0,0,0,0,128,192,224
,224,192,128,0
00250 DATA 0,0,0,0,1,1,3,7,15,15,15,15,1
5,31,24,0
00260 DATA 0,0,3,127,255,241,227,199,255
,255,255,255,254,192,0,0
00270 DATA 0,24,248,240,240,240,240,240,
224,192,128,128,0,0,0,0
00280 DATA 0,0,0,0,1,3,3,3,7,15,31,63,16
,8,4,3
00290 DATA 60,36,126,255,255,255,255,255
,255,255,255,24,60,60,0,255
00300 DATA 0,0,0,0,128,192,192,192,224,2
40,248,124,8,16,32,192
00310 DATA 255,128,190,161,161,161,161,1
90,161,161,161,161,161,190,128,255
00320 DATA 255,0,60,66,66,66,66,66,126,6
6,66,66,66,66,0,255
00330 DATA 255,1,249,133,133,133,133,249
,145,145,137,137,133,133,1,255
00340 DATA 0,0,0,0,0,3,7,14,15,6,3,1,0,0
,0,0
00350 DATA 1,115,54,12,24,255,111,253,18
3,254,247,223,253,118,60,0
00360 DATA 128,0,0,0,0,192,96,240,240,22
4,192,128,0,0,0,0
00370 DATA 0,30,63,126,254,255,255,252,2
52,248,120,48,0,0,31,0
00380 DATA 60,122,127,112,113,255,254,25
4,255,127,127,51,17,17,255,0
00390 DATA 0,56,124,254,254,255,255,127,
31,31,15,7,1,0,252,0
00400 FOR I=63488+65*16 TO 63488+86*16-1
00410 READ A:POKE I,A:NEXT I
00420 A1$="ABC":B1$="DEF":C1$="GHI":D1$=
"JKL":E1$="MNO":F1$="PQR":G1$="STU"
00430 Z1$(1)=A1$:Z1$(2)=B1$:Z1$(3)=C1$:Z
1$(4)=D1$:Z1$(5)=E1$:Z1$(6)=F1$
00440 PRINTTAB(6);"Would you like sound
[Y/N] :- ";
00450 INPUT A0$:IF A0$="N"OR A0$="n" THE
N LET B=0
00460 CLS:CURS 14,1:PRINT"*** THE WINNIN
G COMBINATIONS ***"
00470 PRINT\\"  Some combinations are wo
rth more than others.  ie. three"\"of a
kind is worth the most, while if the fir
st and second"\"windows are the same, (o
r the second and thir";
00480 PRINT"d), this is worth"\"more tha
n the first and the third windows the sa
me.  e.g :-"
00490 PRINT\\"All the symbols have an or
der of value, e.g (from left)."
00500 PCG:CURS 7,9:PRINT E1$;" ";E1$;" "
;E1$;:NORMAL:PRINT"    >    ";:PCG:PRINT
 E1$;" ";E1$;"XXX";:NORMAL:PRINT"    >
  ";:PCG:PRINT E1$;"XXX";E1$
00510 CURS 10,12:PRINT E1$;:GOSUB 1580:P
RINT A1$;:GOSUB 1580:PRINT F1$;:GOSUB 15
80:PRINT D1$;:GOSUB 1580:PRINT B1$;:GOSU
B 1580:PRINT C1$:NORMAL
00520 CURS 10,13:PRINT"Bar    Apple Stra
wberry Bell   Pear   Lemon"
00530 CURS 16,15:PRINT"-=*(PRESS ANY KEY
 TO START)*=-"
00540 Q1$=KEY$:IF Q1$=""THEN 540
00550 HIRES:CURS 23,4:PRINT"*** GAMBLER
***"
00560 PLOT 160,60 TO 310,60 TO 310,220 T
O 160,220 TO 160,60
00570 PLOT 175,180 TO 297,180 TO 297,155
 TO 256,155 TO 256,180
00580 PLOT 255,180 TO 255,155 TO 217,155
 TO 217,180
00590 PLOT 216,180 TO 216,155 TO 174,155
 TO 174,180
00600 PLOT 175,180 TO 175,155
00610 PLOT 296,180 TO 296,155:PLOT 160,1
40 TO 310,140
00620 PLOT 190,60 TO 190,140: PLOT 280,6
0 TO 280,140
00630 PLOT 205,96 TO 210,78 TO 258,78 TO
 263,96 TO 205,96
00640 PLOT 173,100 TO 178,100 TO 178,104
 TO 173,104 TO 173,100
00650 PLOT 175,104 TO 175,125:PLOT 176,1
04 TO 176,125
00660 PLOT 175,75 TO 175,100:PLOT 176,75
 TO 176,100
00670 PLOT 176,104 TO 185,125:PLOT 175,1
04 TO 166,125
00680 PLOT 176,100 TO 185,75:PLOT 175,10
0 TO 166,75
00690 PLOT 293,100 TO 298,100 TO 298,104
 TO 293,104 TO 293,100
00700 PLOT 295,104 TO 295,125:PLOT 296,1
04 TO 296,125
00710 PLOT 295,100 TO 295,75:PLOT 296,10
0 TO 296,75
00720 PLOT 296,104 TO 305,125:PLOT 295,1
05 TO 286,125
00730 PLOT 296,100 TO 305,75:PLOT 295,10
0 TO 286,75
00740 PLOT 285,220 TO 290,230 TO 305,230
 TO 310,220
00750 PLOT 288,226 TO 307,226:PLOT 288,2
25 TO 308,225
00760 PLOT 287,224 TO 307,224:PLOT 287,2
23 TO 308,223
00770 PLOT 286,222 TO 308,222:PLOT 286,2
21 TO 309,221
00780 PLOT 310,125 TO 320,125 TO 320,140
 TO 310,140
00790 PLOT 320,129 TO 321,129 TO 321,136
 TO 320,136
00800 GOSUB 1040:GOSUB 1100:GOSUB 1160:G
OSUB 1220
00810 Q=1:GOSUB 1160:GOSUB 1100:GOSUB 10
40
00820 X=INT(RND*7):IF X=0 THEN 820
00830 Y=INT(RND*7):IF Y=0 THEN 830
00840 Z=INT(RND*7):IF Z=0 THEN 840
00850 PCG:CURS 29,9:PRINT G1$
00860 FOR I=1 TO 3:FOR U=1 TO 6
00870 CURS 24,6:PRINT Z1$(U):CURS 29,6:P
RINT Z1$(U):CURS 34,6:PRINT Z1$(U):FOR P
=1 TO 80:NEXT P:NEXT U:NEXT I
00880 CURS 24,6:PRINT Z1$(X):PLAY B
00890 FOR I=1 TO 2:FOR U=1 TO 6
00900 CURS 29,6:PRINT Z1$(U):CURS 34,6:P
RINT Z1$(U):FOR P=1 TO 80:NEXT P:NEXT U:
NEXT I
00910 CURS 29,6:PRINT Z1$(Y):PLAY B
00920 FOR I=1 TO 2:FOR U=1 TO 6
00930 CURS 34,6:PRINT Z1$(U):FOR P=1 TO
80:NEXT P:NEXT U:NEXT I
00940 CURS 34,6:PRINT Z1$(Z):PLAY B:NORM
AL
00950 GOSUB 1280
00960 IF M+1>0 THEN CURS 29,11:PRINT"$";
M+1:IF B<>0 THEN PLAY 0,3;11;11;11;11;11
;11;11;11
00970 V=V+M:M=0
00980 CURS 4,14:IF V>9 THEN PRINT"YOUR T
OTAL NOW IS :- $";V ELSE PRINT"YOUR TOTA
L NOW IS :- $";V;" "
00990 CURS 4,15:INPUT"ANOTHER GO ? [Y/N]
:- "R1$
01000 CURS 4,15:IF B=0 AND V=0 THEN PRIN
T"SORRY YOU'RE BROKE !!!!!":END ELSE IF
V=0 THEN PRINT"SORRY YOU'RE BROKE !!!!!"
:PLAY 4,2;2,4:END
01010 IF R1$<>"N" AND R1$<>"n" THEN CURS
 26,14:IF V>9 THEN PRINT V-1;" ":CURS 29
,11:PRINT"    ":GOTO 820
01020 IF R1$<>"N" AND R1$<>"n" THEN CURS
 26,14:IF V<10 THEN PRINT V-1;"  ":CURS
29,11:PRINT"    ":GOTO 820
01030 PRINT"OK! BYE FOR NOW.        ":END
01040 REM ROUTINE TO MOVE HANDLE STAGE 1
01050 PLOT 322,129 TO 322,165 TO 330,175
 TO 330,210
01060 PLOT 323,129 TO 323,165 TO 331,175
 TO 331,210
01070 IF Q<>1 THEN PLOTR 322,129 TO 322,
165 TO 330,175 TO 330,210
01080 IF Q<>1 THEN PLOTR 323,129 TO 323,
165 TO 331,175 TO 331,210
01090 Q=0:FOR T=1 TO 240:NEXT T:RETURN
01100 REM MOVE HANDLE STAGE 2
01110 PLOT 322,129 TO 322,151 TO 331,159
 TO 331,180
01120 PLOT 323,129 TO 323,151 TO 332,159
 TO 332,180
01130 PLOTR 322,129 TO 322,151 TO 331,15
9 TO 331,180
01140 PLOTR 323,129 TO 323,151 TO 332,15
9 TO 332,180
01150 FOR T=1 TO 240:NEXT T:RETURN
01160 REM MOVE HANDLE STAGE 3
01170 PLOT 322,129 TO 322,140 TO 327,146
 TO 327,156
01180 PLOT 323,129 TO 323,140 TO 328,146
 TO 328,156
01190 PLOTR 322,129 TO 322,140 TO 327,14
6 TO 327,156
01200 PLOTR 323,129 TO 323,140 TO 328,14
6 TO 328,156
01210 FOR T=1 TO 240:NEXT T:RETURN
01220 REM MOVE HANDLE STAGE 4
01230 PLOT 322,131 TO 327,131 TO 327,132
 TO 322,132
01240 PLOT 322,133 TO 327,133 TO 327,134
 TO 322,134
01250 PLOTR 322,131 TO 327,131 TO 327,13
2 TO 322,132
01260 PLOTR 322,133 TO 327,133 TO 327,13
4 TO 322,134
01270 FOR T=1 TO 240:NEXT T:RETURN
01280 REM MONEY CHANGE
01290 IF X=Y AND X=Z THEN GOSUB 1340:GOT
O 1330
01300 IF X=Y THEN GOSUB 1420:GOTO 1330
01310 IF Y=Z THEN GOSUB 1500:GOTO 1330
01320 IF X=Z THEN LET M=M+2
01330 M=M-1:RETURN
01340 REM THREE OF A KIND
01350 IF X=5 THEN LET M=M+30
01360 IF X=1 THEN LET M=M+25
01370 IF X=6 THEN LET M=M+21
01380 IF X=4 THEN LET M=M+18
01390 IF X=2 THEN LET M=M+16
01400 IF X=3 THEN LET M=M+15
```

Microbee

```
01410 RETURN
01420 REM 1st & 2nd
01430 IF X=5 THEN LET M=M+7
01440 IF X=1 THEN LET M=M+6
01450 IF X=6 THEN LET M=M+5
01460 IF X=4 THEN LET M=M+4
01470 IF X=2 THEN LET M=M+3
01480 IF X=3 THEN LET M=M+2
01490 RETURN
01500 REM 2nd & 3rd
01510 IF Y=5 THEN LET M=M+7
01520 IF Y=1 THEN LET M=M+6
01530 IF Y=6 THEN LET M=M+5
01540 IF Y=4 THEN LET M=M+4
01550 IF Y=2 THEN LET M=M+3
01560 IF Y=3 THEN LET M=M+2
01570 RETURN
01580 REM SPACER
01590 NORMAL:PRINT"     "j:PCG:RETURN
```

VZ200

VZ-200 CASSETTE INLAYS

This program is for all you VZ-200/300 users who have piles of cassette tapes and want to index their contents so it's easy to find what you want. This program uses the PP-40, a printer/plotter distributed by Dick Smith, and makes extensive use of the graphics command supported by this printer. The program contains comments for those users unfamiliar with the required commands, and for those who are thinking of converting the program.

Ian Dutfield,
Cromer, NSW.

```
5 GOSUB 1000 'TITLE
10 'CASSETTE TAPE INSERTS
20 'BY IAN DUTFIELD
25 'FOR THE VZ-200
30 '   16/3/85
40 'FOR USE WITH PP40
50 'PRINTER
60 'USE IN 40 COLUMN MODE
70 'SET PRINTER TO TEXT MODE
75 ' CAN BE CONVERTED TO OTHER
   PRINTERS.
80 LPRINT CHR$(17)
90 'CR AND LINEFEED
100 LPRINT CHR$(13)
110 LPRINT CHR$(10)
120 'SET COLOUR TO BLACK
130 'FIRST GO INTO GRAPHIC MODE
140 LPRINT CHR$(18)
150 LPRINT "C0"
160 'RETURN TO TEXT
170 LPRINT CHR$(17)
```

SMOOTH TALK YOUR FLOOZY
NO. 1 IN AN OCCASIONAL EDUCATIONAL SERIES FOR HETERO HACKERS

NOTES:

VZ200

```
180 LPRINT " *** CASSETTE INLAYS ***"
190 LPRINT ""
200 ' INTO GRAPHIC MODE TO
210 ' PRINT NUMBERS AND LINES
220 LPRINT CHR$(18)
230 LPRINT "S1"
240 ' SET SIZE
245 ' PRINT NUMBERS
255 LPRINT "P1."
260 'DRAW LINE
270 LPRINT "J446,0"
280 ' GO BACK TO PRINT NUMBER
290 LPRINT "R-200,0"
300 ' PRINT OTHER NUMBER
310 LPRINT "P2."
315 LPRINT"R-292,-30"
320 LPRINT"P3."
321 LPRINT"J446,0"
322 LPRINT"R-200,0"
323 LPRINT"P4."
324 LPRINT"R-292,-30"
325 LPRINT"P5."
326 LPRINT"J446,0"
327 LPRINT"R-200,0"
328 LPRINT"P6."
329 LPRINT"R-292,-30"
330 LPRINT"P7."
340 LPRINT"J446,0"
350 LPRINT"R-200,0"
360 LPRINT"P8."
370 LPRINT"R-292,-30"
380 LPRINT"P9."
390 LPRINT"J446,0"
400 LPRINT"R-200,0"
410 LPRINT"P10."
420 LPRINT"R-315,-30"
430 LPRINT"P11."
440 LPRINT"J446,0"
450 LPRINT"R-200,0"
460 LPRINT"P12."
470 LPRINT"R-315,-30"
```

VZ200

```
480 LPRINT"P13."
490 LPRINT"J446,0"
500 LPRINT"R-200,0"
510 LPRINT"P14."
520 LPRINT"R-315,-30"
920 SOUND 31,1
930 PRINT"(INVERSE) FINISHED":FOR T=1 TO
1500:NEXT:RUN
1000 'TITLE PAGE
1010 CLS
1030 COLOR 8,0
1035 POKE 30744,1
1040 PRINT@0,"CTRL+Q,CTRL+T*30,CTRL+W";
1045 PRINT@448,"CTRL+E,CTRL+Y*30,CTRL+R"
;
1060 FOR Y=32 TO 416 STEP 32
1070 PRINT@Y,"CTRL+U"
1080 NEXT Y
1090 FOR Y=63 TO 447 STEP 32
2000 PRINT@Y,"CTRL+I"
2010 NEXT Y
2040 PRINT@109,"VZ-200"
2050 PRINT@195,"*** CASSETTE - INLAYS **
*"
2060 PRINT@298,"BY IAN DUTFIELD"
2070 PRINT@388,"PRESS ANY KEY TO CONTINU
E"
2080 IF INKEY$="" THEN GOTO 3000
2090 IF INKEY$="" THEN GOTO 3000
2095 SOUND 31,1:GOTO 4000
3000 SOUND 28,1
3010 PRINT@388,"(INVERSE)PRESS ANY KEY T
O CONTINUE"
3020 SOUND 10,1
3030 GOTO 2070
4000 CLS
4005 POKE 30744,0
4010 INPUT"(INVERSE)SET UP PRINTER AND P
RESS <RET>";P$
4020 PRINT:PRINT:PRINT"PRINTING"
4030 RETURN
```

Microbee

HIGH-RES ON SCREEN 2

This program allows you to plot on 'screen 2' of the Microbee, simply by exchanging the data of both screens, and will also handle colour information. It therefore eliminates the need to peek and poke the alternate screen. You can insert your own program after line 720, but my listing incorporates a library of screen utility subroutines. These are mainly BASIC statements, so you can take the program apart at your leisure.

Ian Florence,
Magill, SA.

```
00100  REM HIRES ON SCREEN 2 for the MICROBEE by Ian Florance 1985
00110  REM (it also works on colour bees!)
00120  REM
00130  REM This program eliminates the need to poke, poke and
00140  REM peek to and from the alternate screen.
00150  REM It does this by a machine code subroutine
00160  REM which exchanges the screens.
00170  REM Therefore you can simply use as per normal the usual
00180  REM basic commands.
00190  REM ALSO the program allows you to alter the screen
00200  REM display so that you can see both screens at once.
00210  REM All the screen options use BASIC statements which
00220  REM you may find useful.
00230  REM
00240 GOTO510
00250  REM SUBROUTINE TO LOOK AT SECOND SCREEN
00260 IN#0 OFF:OUT 12,12:OUT 13,4:IN#0 ON:RETURN
00270  REM
00280  REM SUBROUTINE TO LOOK AT FIRST SCREEN
00290 IN#0 OFF:OUT 12,12:OUT 13,0:IN#0 ON:RETURN
00300  REM
00310  REM SUBROUTINE TO GO TO INTERLACE SCAN
00320 IN#0 OFF:OUT 12,8:OUT 13,75:IN#0 ON:RETURN
00330  REM
00340  REM SUBROUTINE TO GO TO NORMAL SCAN
00350 IN#0 OFF:OUT 12,8:OUT 13,72:IN#0 ON:RETURN
00360  REM
00370  REM SUBROUTINE TO EXCHANGE SCREENS AND ENTITLE THEM WITH
00380  REM SCREEN 1 & SCREEN 2
00390 CURS 28,1:COLOR 6:COLORB0:PRINT"SCREEN 2":B=USR(A):COLOR 13:CURS 28,1:PRIN
T"SCREEN 1":COLOR 7:COLORB1:RETURN
00400  REM
00410  REM MACHINE CODE DATA TO EXCHANGE SCREENS
00420  REM        (it will also handle colour as well!)
00430 DATA  58, 0, 248, 71, 62, 0, 50, 0, 248, 62, 64, 211
```

Microbee

```
00440 DATA  8, 58, 0, 248, 50, 0, 248, 79, 62, 0, 211, 8
00450 DATA  58, 0, 248, 254, 0, 120, 50, 0, 248, 32, 34, 62
00460 DATA  64, 211 , 8, 121, 50, 0, 248, 33, 0, 248, 17
00470 DATA  0, 252, 1, 0, 4, 126, 8, 26, 119, 8, 18, 19, 35
00480 DATA  11, 120 , 177, 32, 243, 62, 0, 211, 8, 33, 0
00490 DATA  240, 17 , 0, 244, 1, 0, 4, 126, 8, 26, 119, 8
00500 DATA  18, 35, 19, 11, 120, 177, 32, 243, 201
00510 CLEAR:SD8:STRS(768):C=0:DIM E(256),F(256):A=PEEK(161)*256-2048
00520  REM Machine code is completely relocatable
00530  REM I have chosen 2k down from top of memory
00540 FORI=ATOA+91:READ B:C=B+C:POKE I,B:NEXTI
00550 IF C<>7314THEN CLS:PRINT:PRINT"ERROR IN MACHINE CODE DATA - CHECKSUM ERROR
":STOP
00560  REM clear both screens
00570 COLOR 7:COLORB4:CLS:PRINTTAB(28);"SCREEN 2":B=USR(A):COLORB1:CLS:PRINTTAB(
28);"SCREEN 1"
00580  REM MENU
00590 A1$="PRESS '1'- view screen 1, '2'- view screen 2, '3'- normal scan,
 '4'- exchange screens 1 & 2, "
00595 A1$=A1$+"'5'- interlace scan mode on,         any other key to continue"
00600 C=LEN(A1$):FORI=1TOC:E(I)=PEEK(61503+I):POKE 61503+I,ASC(A1$(;I,I)):NEXTI
00610 REM next line is to poke in color if necessary
00620 IF PEEK(153)=255THENFORI=1TOC:OUT 8,64:F(I)=PEEK(63551+I):POKE 63551+I,7:N
EXTI:OUT 8,0
00630 A1$=KEY$:IFA1$=""THEN630
00640 FORI=1TOC:POKE 61503+I,E(I):NEXTI
00650 IFPEEK(153)=255THENFORI=1TOC:OUT 8,64:POKE 63551+I,F(I):NEXTI:OUT 8,0
00660 IF A1$="2"THENGOSUB250:GOTO590
00670 IF A1$="1"THENGOSUB280:GOTO590
00680 IF A1$="5"THENGOSUB310:GOTO590
00690 IF A1$="3"THENGOSUB340:GOTO590
00700 IF A1$="4"THENGOSUB370:GOTO590
00710  REM
00720  REM YOUR program may be inserted from here on
00730  REM My program draws a four-leaf clover and cardiod
```

Microbee

```
00740  REM on screen 2, switches on interlace scan and returns
00750  REM to the menu
00760 B=USR(A):COLORBO:SD4:HIRES:X1=128:Y1=128:X2=384:Y2=128:S1=1.6*30:S2=30
00770  REM x1,y1 is the centre of the first figure & x2,y2 the center of the 2nD
00780  REM s1 & s2 are the scaling factors for the x & y coords
00790 FORF1=0TO6.28STEPO.04:C1=COS(F1):C2=SIN(F1)*S2:R1=1+C1:R2=SQR(ABS(COS(2*F1
))):COLOR 6:SET INT(R1*C1*S1+X1),INT(R1*C2+Y1):COLOR 13:SET INT(R2*C1*S1*2+X2),I
NT(R2*C2*2+Y2):NEXTF1
00800 COLOR 4:CURS 18,2:PRINT"CARDIOD":CURS 40,2:PRINT"FOUR-LEAF CLOVER"
00810 SD8:COLOR 7:COLORB1:B=USR(A):GOSUB310:GOTO590
```

Amstrad

BYTECYCLE SURROUND

Bytecycle Surround is a two-player game in which each player tries to force the other into the energy field surrounding the playing area, or to crash into the trail of a bytecycle.

Player one moves using the joystick, and player two moves using the cursor arrow keys. This arrangement can be changed by altering the key numbers in lines 530 — 560.

Bruce Daniel,
Mudgee, NSW.

```
1 ' Bytecycle Surround
2 '
10 MODE 0:CLS
20 GOSUB 500
30 x1=299:y1=200
40 x2=340:y2=200
50 sx.1=-4:sy.1=0 : x=4 : y=2
60 sx.2=+4:sy.2=0
70 WHILE NOT(P1 AND P2)
80 IF NOT INKEY(uP.1) THEN sy.1=y:sx.1=0
90 IF NOT INKEY(uP.2) THEN sy.2=y:sx.2=0
100 IF NOT INKEY(dw.1) THEN sy.1=-y:sx.1=0
110 IF NOT INKEY(dw.2) THEN sy.2=-y:sx.2=0
120 IF NOT INKEY(lf.1) THEN sx.1=-x:sy.1=0
130 IF NOT INKEY(lf.2) THEN sx.2=-x:sy.2=0
140 IF NOT INKEY(rt.1) THEN sx.1=x:sy.1=0
150 IF NOT INKEY(rt.2) THEN sx.2=x:sy.2=0
160 P1=TEST(x1+sx.1,y1+sy.1) : P2=TEST(x2+sx.2,y2+sy.2)
170 IF P1<>0 OR P2<>0 THEN 300
180 MOVE x1,y1 : x1=x1+sx.1 : y1=y1+sy.1 : DRAW x1,y1,2
190 MOVE x2,y2 : x2=x2+sx.2 : y2=y2+sy.2 : DRAW x2,y2,3
200 WEND
210 '
300 IF P1<>0 THEN P$="Player 2" ELSE IF P2<>0 THEN P$="Player 1"
310 SOUND 2,91,49,14,0,1,1
320 IF P1<>0 AND P2<>0 THEN P$="Nobody"
330 w$=P$+" has Won."
340 LOCATE 10-LEN(w$)\2,2
350 PEN 14:PRINT w$
360 WHILE INKEY$<>"":WEND
370 PEN 15:LOCATE 4,24:PRINT "Press <ENTER>";
380 ky$="":WHILE ky$<>CHR$(13) AND ky$<>"X":ky$=INKEY$:WEND
390 RUN
400 '
500 INK 0,10:INK 1,26:INK 2,6:INK 3,14:BORDER 0
510 MOVE 0,0
520 DRAW 639,0,1:DRAW 639,399:DRAW 0,399:DRAW 0,0
530 uP.1=72 : uP.2=0
540 dw.1=73 : dw.2=2
550 lf.1=74 : lf.2=8
560 rt.1=75 : rt.2=1
570 RETURN
```

VZ200

MORSE TUTOR (again)

In the June '85 issue of *Your Computer* we published a Morse Tutor program written by Basil Heath for the TRS-80 MC10. It was wrongly listed as being intended for the VZ200. As a result, Basil received several letters and phone calls from VZ200 users who pointed out first that the print had been reduced so much it was difficult to read, and second that the program didn't work (for obvious reasons).

Basil has very kindly collaborated with a friend who owns a VZ200 in rewriting his program for that machine. Here we've listed both versions (with the right machine headings, this time). Our apologies to the many people we mislead by this mistake.

```
1 REM MORSE TUTOR PROGRAM:CLS
2 PRINT"AUTO-RUN":PRINT"YES(1)":PRINT"NO(2)"
4 CLEAR 500
5 DATA 63,62,60,56,48,32,33,35,39,47,6,17,21,9,2,20,11,16,4,30
6 DATA 13,18,7,5,15,22,27,10,8,3,12,24,14,25,29,19
7 INPUTR:DIMB$(36)
8 FORI=1TO36:READJ:LETB$(I)=CHR$(J)
9 NEXTI:CLS
10 INPUT"SPEED(WPM)(MAX 10)?";SPEED
11 LET SPEED=5.0/SPEED
12 INPUT"DELAY(0-15)?";DELAY:DELAY=DELAY*50
13 INPUT"NO:-CHARACTERS(MAX 200)";N
14 INPUT"LETTERS(1)NUMBERS(2)OR BOTH(3)?";L
15 DIM T$(N)
16 CLS:PRINTTAB(5)"MORSE TUTOR PROGRAM":FORI=1TON
17 LETT$(I)=CHR$(RND(10*-1*(L<>1)+26*-1*(L<>2))+10*-1*(L=1))
18 NEXTI:FORI=1TON
19 LETX=ASC(B$(ASC(T$(I))))
20 GOSUB65
23 IF I=INT(I/5)*5THEN29
25 IFI=NTHEN32
27 NEXTI
29 FOR Z=1 TO INT(200*SPEED+(DELAY*5)):NEXTZ
30 GOTO25
32 FORI=1TON
34 IF ASC(T$(I))>10 THEN39
37 PRINTCHR$(ASC(T$(I))+47);
38 GOTO40
39 PRINTCHR$(ASC(T$(I))+54);
40 IF I=INT(I/25)*25THEN46
41 IF I=INT(I/5)*5THEN44
42 IF I=NTHEN49
43 NEXTI
44 PRINT" ";
45 GOTO42
46 PRINT
47 GOTO42
49 IF R<>2THEN90:PRINT:PRINT:PRINT
50 PRINT"PRESS KEY(1)(ENTER) TO RE-TRY"
51 PRINT"PRESS KEY(2)(ENTER) TO EXIT"
52 INPUTP:IFP<2THEN16
53 DATA 80,82,79,71,82,65,77,32,66,89,58,45,32,66,46,72,69,65
54 DATA 84,72,32,86,75,52,65,66,72
55 CLS:PRINT:PRINT
57 FORI=1TO27
59 READA
61 PRINT CHR$(A);
63 NEXTI:END
65 LETY=X/2:LETX=INT(Y)
67 Q=(2*SPEED*(1+(Y-X)*4))
70 SOUND25,Q
75 IFX=1THEN80
77 FORZ=1TOINT(40*SPEED):NEXTZ
78 GOTO65
80 FORZ=1TOINT(120*SPEED+(DELAY*3)):NEXTZ
85 RETURN
90 PRINT:PRINT:PRINT"PRESS-'BREAK'-TO EXIT".
95 FORI=1TO10000:NEXTI:GOTO16
```

TRS-80 MC10

```
1 CLS
2 PRINT"AUTO-RUN":PRINT"YES(1)":PRINT"NO-(2)"
4 CLEAR 500
5 DATA 63,62,60,56,48,32,33,35,39,47,6,17,21,9,2,20,11,16,4,30,13,18,7,5,15,22,2
7,10,8,3,12,24,14,25,29,19
6 INPUT R
7 DIM B$(36)
```

TRS-80 MC10

```
8 FOR I=1 TO 36:READ J:LET B$(I)=CHR$(J)
9 NEXT I:CLS
10 INPUT"SPEED(WPM)(MAX 15)?";SPEED
11 LET SPEED=7.5/SPEED
12 INPUT"DELAY(0 15)?";DELAY:DELAY=DELAY*50
13 INPUT"NO. CHARACTERS(MAX 200)";N
14 INPUT"LETTERS(1)NUMBERS(2)OR BOTH(3)?";L
15 DIM T$(N)
16 CLS:PRINT TAB(5)"MORSE TUTOR PROGRAM":FOR I=1 TO N
17 LET T$(I)=CHR$(RND(10*-1*(L<>1)+26*-1*(L<>2))+10*-1*(L=1)):NEXT I
18 FOR I=1 TO N
19 LET X=ASC(B$(ASC(T$(I))))
20 GOSUB 65
23 IF I=INT(I/5)*5 THEN 29
25 IF I=N THEN 32
27 NEXT I
29 FOR Z=1 TO INT(200*SPEED+(DELAY*5)):NEXT Z
30 GOTO 25
32 FOR I=1 TO N
34 IF ASC(T$(I))<10 THEN 39
37 PRINT CHR$(ASC(T$(I))+47);
38 GOTO 40
39 PRINT CHR$(ASC(T$(I))+54);
40 IF I=INT(I/25)*25 THEN 46
41 IF I=INT(I/5)*5 THEN 44
42 IF I=N THEN 49
43 NEXT I
44 PRINT" ";
45 GOTO 42
46 PRINT
47 GOTO 42
49 IF R<>2 THEN 90
50 PRINT:PRINT:PRINT
51 PRINT"PRESS KEY(1)(ENTER)TO RE-TRY":PRINT "PRESS KEY(2)(ENTER)TO EXIT"
52 INPUT P:IF P=2 THEN 16
53 DATA 80,82,79,71,82,65,77,32,66,89,58,45,32,66,46,72,69,65,84,72,32,86,75,52,
65,66,72
54 CLS:PRINT:PRINT
57 FOR I=1 TO 27
59 READ A
61 PRINT CHR$(A);
63 NEXT I:END
65 LET Y=X/2:LET X=INT(Y)
67 Q=(2*SPEED*(1+(Y=X)*4))
70 SOUND 200,Q
75 IF X=1 THEN 80
77 FOR Z=1 TO INT(40*SPEED):NEXT Z
78 GOTO 65
80 FOR Z=1 TO INT(120*SPEED+(DELAY*3)):NEXT Z
85 RETURN
90 PRINT:PRINT:PRINT"PRESS 'BREAK' TO EXIT"
95 FOR I=1 TO 10000:NEXT I:GOTO 16
```

Microbee

DRAWER

Drawer lets you draw high-res pictures on the screen and save them on disk as graphics files. A default extension of .GRF is given to these files (for example, 'picture.grf' may be a file on the directory).

The main menu gives the options:

```
00090 ON ERROR GOTO 999
00100 CLS:PRINT TAB(26)"## DRAWER ##":PRINTTAB(26)"##############"
00105 CURS 28,6:PRINT">>MENU<<"
00110 CURS 26,8:PRINT"1] DRAW":PRINTTAB(26)"2] LOAD FILE":PRINTTAB(26)"3] KILL F
ILE":PRINT TAB(26)"4] DIRECTORY":PRINTTAB(26)"5] END"
00120 A1$=KEY$:IF A1$="" THEN 120
00130 LET A2=VAL(A1$):LET B=INT(A2)
00140 ON B GOTO 200,400,600,800,850
00150 GOTO 120
00200 CLS:PRINT "  To create your drawing use the keys"\\TAB(35)"Q   W   E"\TAB(
35)"A       D"\TAB(35)"Z   X   C"
00210 PRINT\"Each key is used to direct the pointer (pencil).                '1'
 is used to turn pointer on & '2' off."
```

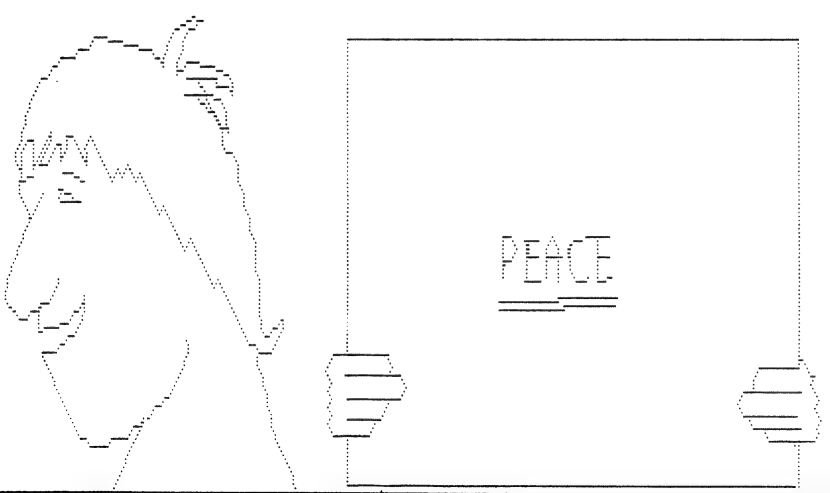
Microbee

1) draw a picture
2) load a graphics file
3) kill a file
4) display the directory or
5) end the program

The program will give instructions if you select option 1. The keys Q, W, E, A, D, Z, X, C are used to guide the pointer around the screen. Typing '2' turns the pointer off, allowing you to move around the screen without drawing lines, as well as letting you erase old lines. The DRAW mode also allows you to save the picture and continue. The number of PCGs used is shown in the top left corner — this must not exceed 128.

Ken Rowe
Dernancourt, SA

```
00220 PRINT\"To save your drawing on a file press 'O'.                    To
return to the menu press 'M'."\
00222 PRINT"The number shown in the top left corner can NOT exceed 128"\\TAB(37)
"Press space bar to start...";
00230 A2$=KEY$: IF A2$<>" " THEN 230
00240 CLS:HIRES:LET X=240:Y=122:V=0
00245 CURS1:PRINT"                          DRAW MODE"
00248 FOR I=0 TO 481 STEP 480:FOR J=0 TO 238 STEP 8:SET I,J:NEXT J:NEXT I
00250 SET X,Y
00255 IF V=1 THEN RESET X,Y
00260 LET Z=PEEK(258): IF Z=1 THEN LET X=X-1 ELSE IF Z=4 THEN LET X=X+1 ELSE IF
Z=23 THEN LET Y=Y+1 ELSE IF Z=24 THEN LET Y=Y-1
00270 IF Z=17 THEN LET X=X-1:Y=Y+1 ELSE IF Z=5 THEN LET X=X+1:Y=Y+1 ELSE IF Z=26
 THEN LET X=X-1:Y=Y-1 ELSE IF Z=3 THEN LET X=X+1:Y=Y-1
00272 IF X>480 THEN LET X=480 ELSE IF X<1 THEN LET X=1
00273 IF Y>238 THEN LET Y=238 ELSE IF Y<1 THEN LET Y=1
00275 IF Z=33 THEN LET V=0 ELSE IF Z=34 THEN LET V=1
00277 IF Z=13 THEN 100
00280 IF Z=32 THEN 310
00290 IF USED<128 THEN CURS1:PRINT USED
00295 IF USED=128 THEN CURS1:PRINT"STOP !!":OUT 2,59:OUT 2,65
00300 GOTO 250
00310 CURS1:PRINT"                                        ":CURS1:INPUT"NAME ?":N1$
;
00320 IF LEN(N1$)>8 THEN 310                                  '• .
00322 CURS1:PRINT"FILE SAVED AS '";N1$;".GRF'"
00325 LET N1$=N1$+".GRF"
00330 CLOSE 6: OPEN "A",6,N1$
00340 GRSAVE 6
00350 CLOSE 6
00360 GOTO 245
00400 CLS: CLOSE 6
00410 INPUT "WHAT FILENAME ?";N2$: N2$=N2$+".GRF"
00420 OPEN "I",6,N2$
00440 GRLOAD 6
00470 CLOSE 6
00480 CURS1:LETX=256:Y=128:V=1:GOTO 245
00600 CLS:DIR:PRINT\\:INPUT"FILE TO KILL ?";N3$: LET N3$=N3$+".GRF"
00610 KILL N3$
00620 GOTO 100
00800 CLS:DIR
00805 PRINT\\"PRESS SPACE BAR TO RETURN TO MENU"
00810 IF KEY$<>" " THEN 810 ELSE 100
00850 CLS:PRINT"END":END
00999 PRINT""
01000 IF ERRORC=16 THEN CLS:CURS18:INVERSE:PRINT"YOU EXCEEDED 128":PLAY0,15:NORM
AL:GOTO 90
01010 IF ERRORC=45 THEN CLS:CURS19:INVERSE:PRINT"FILE NOT FOUND ERROR":NORMAL:PL
AY0,15:GOTO 90
01020 PRINT"ERROR ";ERRORC;" IN LINE ";ERRORL: END
```



Microbee

SMAKER

Smaker is a 'sprite generator' written in BASIC. It allows you to create a line drawing of something and, by writing down the data, recreate it in your games and so on. It is better than the usual method in that it uses about one-seventh of the memory usually required and is more liquid; that is, you can easily move, scale or rotate the sprite.

To apply the program, first invent a name for your sprite; for example, Maurice.

```
00100 REM SMAKER 25/04/85 RICHARD LARKIN.

00110 ON ERROR GOTO 750 : POKE 162,30 : POKE 163,128 : POKE 257,2

00120 DIM X0(200),Y0(200),K0(1)

00130 P=0 : M=0 : X0(0)=255 : Y0(0)=255 : S0$="                    " : S1$="FBMELSC"
 : S2=9 : CLS : PRINT "SPRITE MAKER-by Richard Larkin."\"Please enter title of s
prite." : I6$="KEY"

00140 ZONE16 : INPUT T1$ : T1$=T1$(;1,16) : PRINT "Would you like the origin,"\"
1) Bottom Corner"\"2) Middle"\"of the grafix screen ?"

00150 O1$=KEY : IF O1$="1" OR O1$="2" THEN 160 ELSE 150

00160 O0=0 : IF O1$="2" THEN LET O0=128

00170 GOSUB 660
```

▶

NOTES:

Microbee

Run the program. Enter the name. Now decide what Maurice looks like and whether you wish to use a co-ordinate system or move a dot about the screen. Some instructions are supplied from within the program.

In KEY entry mode the prompt asks you to enter the co-ordinates of the next point. You may type numbers between 0 and 253 or -128 and 126, depending on location of origin. First type the X co-ordinate followed by either RET or /, then the Y co-ordinate followed by RET or /. The new point will be entered at the present location in the file, and a new line drawn on the screen.

In DOT entry a point is moved about the screen using the keys (, ?, ' and +, which form a diamond on the right side of the keyboard. Selecting the number keys 1-9 will alter the step size taken by the dot. Type P to enter a point when you have arrived at the intended location.

I starts a new line

C changes between DOT and KEY modes

E ends the file at the location of the pointer

B and F take you backwards and forwards through the file

S redraws the sprite with any corrections you have made

M gives you the menu for extra commands

The six extra commands execute movements along the X and Y axes, scale changes of the X and Y axes, rotate sprites, and print data. These are easy to work out, so explanations are not required.

When using the rotate command, the program will ask you how many degrees you wish to turn by, then return you to your input mode to choose the point about which to rotate. Rotating cause scale changes, since the little points on the screen are really rectangles, so a scale correction may be necessary afterwards.

The subroutine allows you to use your data in your own programs, or you might prefer the one I used in the actual program. It's up to you!

Richard Larkin
Dee Why, NSW

```
00180 CURS 0 : PRINT "(SPRITE MAKER):"TI$\"M for menu"\"B back one point"\"F for
ward a point"\"E end sprite"\"L line end"\"S sprite update"\"C change dot/key in
put"

00190 CURS 1,11 : PRINT S0$ : CURS 1,11 : IF X0(P)=255 THEN PRINT "END OF SPRITE
"\S0$ ELSE IF X0(P)=254 THEN PRINT "END OF LINE"\S0$ ELSE PRINT"X-VAL="X0(P)-D0"
  "\"Y-VAL="Y0(P)-D0"   "

00200 PRINT "POINTER"P" "\"BUFFER MAX"M"  "\"PCG USED"USED"  " : IF I6$="DOT" TH
EN 760

00210 K0$(0)="" : K0$(1)="" : CURS 1,9 : PRINT "ENTER POINT XXX/ YYY"[A8 32]\[A2
0 32]\[A9 8]; : FOR X=0 TO 1 : K2$=""

00220 K1$=KEY : IF K1$="" THEN 220

00230 A=ASC(K1$) : IF NOT(A=8 OR A=127) THEN 240 ELSE IF LEN(K2$)0 THEN LET K2$
=K2$(;1,LEN(K2$)-1) : PRINT CHR(127); : GOTO 220 ELSE 220

00240 IF LEN(K2$)=0 THEN IF K1$()"+" AND K1$()"-" THEN PRINT " "; : K2$=" "

00250 IF SEARCH(S1$,K1$)()0 THEN NEXT+X 350

00260 IF K1$="/" OR ASC(K1$)=13 THEN LET K0$(X)=K2$ : GOTO 290

00270 IF (K1$("0" OR K1$)"9") AND K1$()"-" THEN 280 ELSE LET K2$=K2$+K1$ : IF LE
N(K2$)=5 THEN LET K2$=K2$(;1,4) ELSE PRINT K1$;

00280 GOTO 220

00290 CURS 16,10 : PRINT"/"; : NEXT X : X1=VAL(K0$(0))+D0 : Y1=VAL(K0$(1))+D0 :
IF X1)253 OR X1(0 OR Y1)255 OR Y1(0 THEN PLAY 24 : GOTO 210

00300 IF R0=1 THEN RETURN ELSE LET X0(P)=X1 : Y0(P)=Y1

00310 P=P+1 : IF P)M THEN LET M=P : X0(M)=255 : Y0(M)=255

00320 IF P=200 THEN PRINT \"FILE FULL"S0$ : PLAY 24; : I=USR(32774) : P=199

00330 IF X0(P-2)()254 AND X0(P-1)()254 AND P()1 THEN PLOT INT(X0(P-2))+255,INT(Y
0(P-2)) TO INT(X0(P-1))+255,INT(Y0(P-1))

00340 GOTO 190

00350 IF K1$="B" THEN IF P)0 THEN LET P=P-1

00360 IF K1$="F" THEN IF P(M THEN LET P=P+1

00370 IF K1$="E" THEN LET X0(P)=255 : Y0(P)=255 : M=P

00380 IF K1$="L" THEN LET X0(P)=254 : Y0(P)=255 : GOTO 310

00390 IF K1$="S" THEN 170

00400 IF K1$()"C" THEN 410 ELSE IF I6$="DOT" : I6$="KEY" : GOTO 170 ELSE LET I6$
="DOT" : GOTO 170

00410 IF K1$()"M" THEN 190

00420 GOSUB 660

00430 CURS 0 : A=0 : J=0 : PRINT "1) LEFT/RIGHT"\"2) UP/DOWN"\"3) X LARGE/SMALL"
\"4) Y LARGE/SMALL"\"5) ROTATE"\"6) PRINT VALUES"\"7) RETURN TO SPRITE"\"WHICH D
O YOU WISH ? "

00440 B=ASC(KEY)-48 : IF B(1 OR B)7 THEN 440 ELSE IF B=7 THEN HIRES : GOTO 180

00450 ON B GOSUB 470,500,530,560,590,620

00460 GOTO 420

00470 INPUT "HOW FAR RIGHT?" I0

00480 FOR X=0 TO M-1 : IF X0(X)()254 THEN LET X0(X)=X0(X)+I0
```

Microbee

```
00490 NEXT X : GOSUB 710 : IF A=1 THEN 480 ELSE RETURN

00500 INPUT "HOW FAR UP?" I0

00510 FOR X=0 TO M-1 : IF X0(X)<>254 THEN LET Y0(X)=Y0(X)+I0

00520 NEXT X : GOSUB 710 : IF A=1 THEN 510 ELSE RETURN

00530 INPUT "X-SCALE FACTOR ?" I0

00540 FOR X=0 TO M-1 : IF X0(X)<>254 THEN LET X0(X)=(X0(X)-00)*I0+00

00550 NEXT X : GOSUB 710 : IF A=1 : I0=-1/I0 : GOTO 540 ELSE RETURN

00560 INPUT "Y-SCALE FACTOR ?" I0

00570 FOR X=0 TO M-1 : IF X0(X)<>254 THEN LET Y0(X)=(Y0(X)-00)*I0+00

00580 NEXT X : GOSUB 710 : IF A=1 : I0=-1/I0 : GOTO 570 ELSE RETURN

00590 INPUT "DEGREES CLOCKWISE" I0 : R0=1 : S1$="" : IF I6$="DOT" THEN GOSUB 760
 ELSE GOSUB 190
00600 R0=0 : S1$="FBMELSC" : A0=-I0*.01745329 : S7=SIN(A0) : C7=COS(A0) : FOR X=
0 TO M-1 : IF X0(X)<>254 : T0=X0(X)-X1 : X0(X)=C7*T0-S7*(Y0(X)-Y1)+X1 : Y0(X)=C7
*(Y0(X)-Y1)+S7*T0+Y1

00610 NEXT X : GOSUB 710 : IF A=1 THEN 600 ELSE RETURN

00620 A=0 : B=0 : FOR X=0 TO M-1 : IF B<>0 AND A=0 : I=USR(32774)

00630 IF A=0 THEN CLS : CURS 5 : UNDERLINE : PRINT "X-VALUE   Y-VALUE"; : NORMAL
 : B=1 : A=14

00640 IF X0(X)=254 THEN PRINT ,"254 END OF LINE"; : A=A+1 ELSE PRINT ↔18 INT(X0
(X)+.5)↔18 INT(Y0(X)+.5)↔

00650 A=A-1 : NEXT X : PRINT ,"255 END OF DATA"\"    ANY KEY TO CONTINUE" : I=US
R(32774) : RETURN
00660 HIRES : C=0
00670 A=INT(X0(C)) : B=INT(Y0(C)) : IF A=255 OR A=254 THEN RETURN

00680 C=C+1 : X=INT(X0(C)) : IF X=255 THEN RETURN ELSE IF X=254 THEN LET C=C+1 :
 GOTO 670

00690 Y=INT(Y0(C)) : PLOT A+255,B TO X+255,Y
00700 A=X : B=Y : GOTO 680

00710 A=0 : FOR X=0 TO M-1 : IF X0(X)<0 OR X0(X)>255 OR Y0(X)<0 OR Y0(X)>255 : A
=1
00720 IF (INT(X0(X))=255 OR INT(X0(X))=254) AND INT(Y0(X))<>255 : A=1 : X0(X)=X0
(X)+.000001

00730 NEXT X : IF A=1 : J=J+1 : PRINT "SORRY, CHANGE ABORTED."\"AS SPRITE WENT O
FF EDGE !" : PLAY 24;0,5 : I0=-I0 : IF J=2 THEN ERROR

00740 RETURN
00750 ON ERROR GOTO 750 : CURS 1,14 : PRINT "SPRITE TO COMPLEX TO DRAW."\"MAYBE
TRY SMALLER SCALE?" : PLAY 4;5;3 : I=USR(32774) : CLS : GOTO 430

00760 INVERT INT(X1)+255,INT(Y1) : A=PEEK(258) : IF A=27 :Y1=Y1+S2 ELSE IF A=47
 : Y1=Y1-S2 ELSE IF A=8 : X1=X1+S2 ELSE IF A=43 : X1=X1-S2

00770 IF X1<0 : X1=0 ELSE IF X1>253 : X1=253 ELSE IF Y1<0 : Y1=0 ELSE IF Y1>255
 : Y1=255

00780 INVERT INT(X1)+255,INT(Y1) : K1$=KEY : IF SEARCH(S1$,K1$)<>0 THEN 350

00790 IF K1$(":" AND K1$)"0" : S2=VAL(K1$) ELSE IF K1$<>"P" THEN 800 ELSE IF R0=
1 THEN RETURN ELSE 300

00800 CURS 1,9 : PRINT "DOT POS ="INT(X1-00)"/"INT(Y1-00)↑A10 32↑\"STEP SIZE"S2
 : GOTO 760
```

Microbee

```
10000 REM POKE DATA INTO MEMORY

10010 REM LET E=ADDRESS OF DATA FOR SPRITE TO DRAW

10020 REM LET X,Y = CO-ORDINATES TO PLOT SPRITE AT

10030 REM GOSUB TO THIS ROUTINE TO PLOT SPRITE

10040 X=X-128 : Y=Y-128

10050 A=PEEK(E) : E=E+1 : B=PEEK(E)

10060 E=E+1 : C=PEEK(E) : IF C=255 THEN RETURN ELSE IF C=254 THEN LET E=E+1 : GO
TO 10050

10070 E=E+1 : Y=PEEK(E) : ON ERROR GOTO 10080 : PLOT A+X,B+Y TO C+X,D+Y : GOTO 1
0090

10080 ON ERROR GOTO 10090 : PLOT C+X,D+Y TO A+X,B+Y

10090 A=C : B=D : GOTO 10060
```

Microbee

POCKET PUZZLE

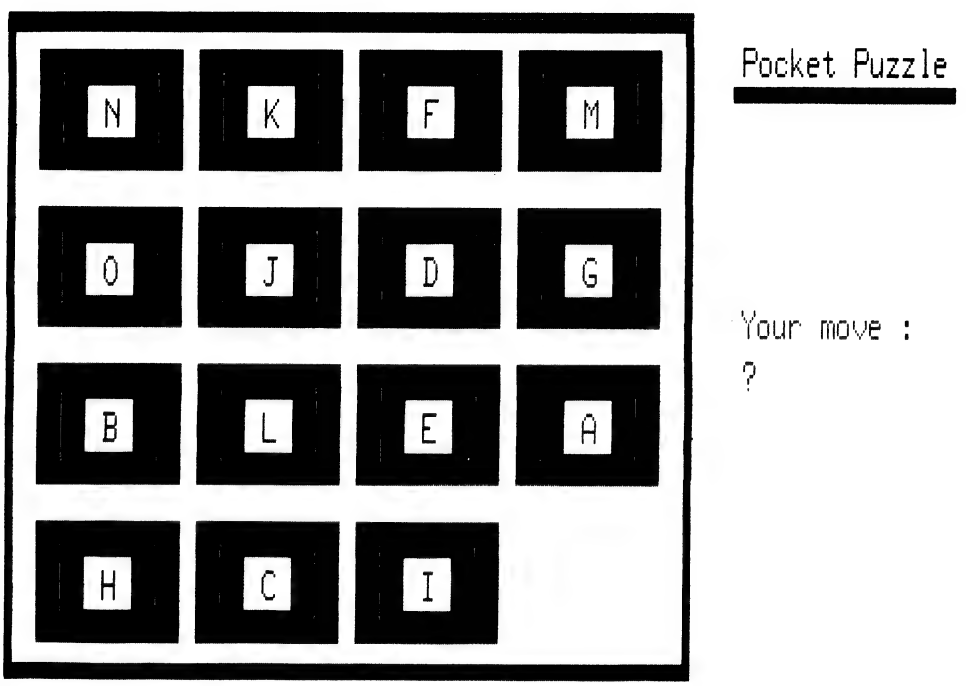
As a child you probably played a handheld version of Pocket Puzzle. In this game the computer jumbles up the letters 'A' to 'O' and leaves you the task of sorting them into alphabetical order. The 15 letters are contained within a 4 by 4 square and only those adjacent to the blank square can be moved.

Lonnie Riley
Banyo, Qld

```
00100 REM ***** Pocket Puzzle *****
00110 CLEAR : CLS : LORES : DIM P(16), G1(100) : F=16
00120 POKE 257,2
00130 H1$=KEY$
00140 INPUT "Do you want instructions (Y/N)? "H1$
00150 IF H1$="Y" THEN GOSUB 640
00160 CLS
00170 REM ***** Create Puzzle *****
00180 RESTORE 700
00190 PLOT 0,45 TO 85,45 TO 85,8 TO 0,8 TO 0,45
00200 CURS 110 : PRINT "Pocket Puzzle" : PLOT 91,41 TO 118,41
00210 FOR X=1 TO 15 : READ A,B,C : FOR Y=1 TO 9 : POKE 60000+A+Y,188 : POKE 6000
0+B+Y,255 : POKE 60000+C+Y,143 : NEXT Y : NEXT X
00220 REM ***** Jumble up the letters *****
00230 FOR X=1 TO 16 : P(X)=0 : NEXT X
00240 FOR X=1 TO 15
00250 Y = INT(RND*15) + 1
00260 IF P(Y) > 0 THEN 250
00270 P(Y)=X
00280 NEXT X
00290 REM ***** Display letters on blocks *****
00300 FOR X=4 TO 12 STEP 4
00310 Y=133+(X/4*192)-192
00320 CURS Y : PRINT " "CHR(P(X-3)+64)" " : CURS Y+10 : PRINT " "CHR(P(X-2)+64)"
 "
00330 CURS Y+20 : PRINT " "CHR(P(X-1)+64)" " : CURS Y+30 : PRINT " "CHR(P(X)+64)
" "
00340 NEXT X
00350 FOR X=10 TO 30 STEP 10 : CURS 699+X : PRINT " "CHR(P(12+X/10)+64)" " : NEX
T X
00360 REM ***** Accept Input *****
00370 CURS 430 : PRINT "Your move : "
00380 CURS 494 : PRINT [A17 32] : PLAY 1,1 : CURS 494 : INPUT G1$
00390 CURS 558 : PRINT [A17 32]
00400 Y=ASC(G1$) : Y=Y-64
00410 FOR X=1 TO 16
00420 IF P(X)=Y THEN LETM=X
00430 NEXT X
00440 IF (F=M+1) OR (F=M-1) OR (F=M+4) OR (F=M-4) THEN 470
00450 CURS 558 : PRINT "Sorry! Try again." : GOTO 380
00460 REM ***** Blank out square to be moved *****
00470 RESTORE 700 : FOR X=1 TO M : READ A,B,C : NEXT X
00480 FOR X=1 TO 9 : POKE 60000+A+X,32 : POKE 60000+B+X,32 : POKE 60000+C+X,32 :
```

Microbee

```
 NEXT X
00490 REM ***** Display square in new position *****
00500 RESTORE 700 : FOR X=1 TO F : READ A,B,C : NEXT X
00510 FOR X=1 TO 9 : POKE 60000+A+X,188 : POKE 60000+B+X,255 : POKE 60000+C+X,14
3 : NEXT X
00520 POKE 60000+B+4,32 : POKE 60000+B+5,P(M)+64 : POKE 60000+B+6,32
00530 P(F)=P(M) : P(M)=0 : F=M
00540 REM ***** Determine if puzzle solved *****
00550 Y=0 : FOR X=1 TO 15
00560 IF P(X)<>X THEN LETY=1
00570 NEXT X
00580 IF Y=0 THEN CURS 812 : PRINT "Congratulations!" : PLAY 1;2;3;4;3;2;1 : GOT
O 600
00590 GOTO 380
00600 H1$=KEY$
00610 CURS 881 : INPUT "Play again? "H1$
00620 IF H1$="Y" THEN 110 ELSE END
00630 REM ***** Instructions *****
00640 PRINT : PRINT "Welcome to Pocket Puzzle!"
00650 PRINT : PRINT "This is the same Pocket Puzzle that you have probably"\"pla
yed as a child."\
00660 PRINT "The computer will mix up the letters A to O and it is your job"\"to
 sort them into alphabetical order. Only letters"\"ajacent to the blank square c
an be moved!"
00670 PRINT : PRINT
00680 PLAY1 : INPUT "Press any key to continue "H1$ : RETURN
00690 REM ***** Data statements for graphics *****
00700 DATA 1505,1569,1633,1515,1579,1643,1525,1589,1653,1535,1599,1663
00710 DATA 1697,1761,1825,1707,1771,1835,1717,1781,1845,1727,1791,1855
00720 DATA 1889,1953,2017,1899,1963,2027,1909,1973,2037,1919,1983,2047
00730 DATA 2081,2145,2209,2091,2155,2219,2101,2165,2229,2111,2175,2239
```

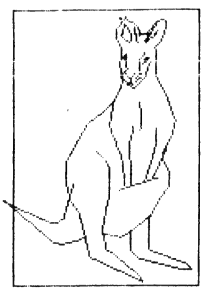


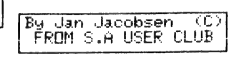
Sega

STAR WAR

This program is for the Sega home computer. It will also work on the Sega 3000 and 3000-H, and can be run from a disk.

Jan Jacobsen
O'Sullivan Beach, SA






```
U
10 REM ******************************
20 REM ***           BY             ***
30 REM ***      JAN JACOBSEN        ***
40 REM ***        (C) 1985          ***
50 REM ******************************
60 SCREEN 2,2:COLOR1,15,,15:CLS
70 RESTORE
80 COLOR1,10,(0,0)-(255,191),10
90 COLOR1,6,(136,8)-(224,60),10
100 COLOR1,6,(130,168)-(255,191),10
110 LINE(130,168)-(255,191),1,B
120 LINE(143,80)-(247,152),1,B
130 LINE(136,8)-(231,60),1,B
140 COLOR1,15,(148,80)-(240,152),10
150 LINE(8,8)-(120,176),1,B
160 COLOR1,15,(8,8)-(115,176),10
170 CIRCLE(88,39),1,1,,,,BF
180 CIRCLE(75,38),1,1,,,,BF
190 CIRCLE(198,106),1,1,,,,BF
200 CIRCLE(206,106),1,1,,,,BF
210 CIRCLE(202,110),2,1,,,,BF
220 CURSOR134,170:PRINT"By Jan Jacobsen  (C)
230 CURSOR134,179:PRINT" FROM S.A USER CLUB"
240 R=R+1
250 READX,Y
260 IFX=0THENREADX,Y:LINE(X,Y)-(X,Y)
270 IFX=-1THEN530
280 LINE-(X,Y)
290 GOTO240
300 DATA0,0,76,12,70,20,71,28,73,32,73,34,72,40,73,44,74,50,72,56,68,64,60,72,52
,78,46,88,42,96,40,100,40,112,42,120,40,128,38,136,35,138,32,138
310 DATA 2,124,32,150,36,151,40,150,44,149,48,148,52,146,55,159,56,160,86,174,84
,170,80,166,68,156,64,154,62,145
320 DATA 72,147,79,143,80,156,84,160,108,166,116,166,112,162,90,154,90,136,92,13
2,96,124,102,116,102,112,102,108,100,104,98,100,98,96,100,92,102,88
330 DATA 104,84,105,80,106,76,104,72,100,60,94,48,94,40,93,32,97,28,99,24,100,20
,99,14,94,16,92,18,88,23,84,25,83,20,80,16,76,13,76,12
340 DATA 0,0,44,114,42,120,42,123,48,132,52,146,0,0,58,94,60,96,64,100,66,104,66
,108,66,120,64,128,62,140,62,145
350 DATA 0,0,75,17,74,20,73,24,76,28,78,29,79,28,79,24,77,20,75,17,0,0,81,19,82,
24,83,29
360 DATA ,0,84,64,85,68,85,72,85,76,84,81
370 DATA 0,0,94,17,92,20,89,24,88,29,0,0,92,20,90,24,87,28,93,25,95,21,0,0,87,35
,84,39,84,45,0,0,76,36,77,44,74,47,74,49,80,55,88,48,91,40
380 DATA 0,0,81,47,79,49,79,51,79,49,76,47,0,0,77,53,78,52,79,52,80,53,0,0,90,38
,86,41,0,0,85,47,82,49,0,0,85,49,82,51
390 DATA 0,0,68,64,66,76,68,84,70,88,71,92,73,96,74,100,76,116,0,0,80,79,78,88,7
9,100,79,112,78,116,0,0,94,80,93,92,91,100,87,112,0,0,100,92,91,112,0,0,79,143,9
0,134,0,0,95,111,72,118,0,0
400 DATA 164,86,207,144,207,148,0,0,168,84,188,110,0,0,184,104,186,96,185,85,0,0
,186,94,182,90,181,85,0,0,178,105,167,97,0,0,171,103,165,104,161,101,0,0,196,120
,204,132,0,0,210,138,216,144,217,148,0,0
410 DATA 213,140,216,138,220,132,221,128,220,124,213,112,211,108,211,105,214,104
,216,102,216,100,212,98,209,98,204,96,200,97,196,98,192,98,187,100,187,104,193,1
08,193,114,0,0
420 DATA 187,111,192,114,194,115,204,115,210,107,210,104,208,100,212,100,213,101
,212,102,210,103,0,0
```

Sega

```
430 DATA 185,117,190,117,197,122,204,120,208,118,0,0,213,134,209,138,204,137,202
,137,0,0,216,128,212,125,208,124,204,126,204,132,206,134,204,132,202,133,200,134
,200,135,204,135,0,0
440 DATA 193,110,186,110,0,0,204,115,200,112,197,109,192,106,196,103,192,101,191
,103,193,106,0,0,186,111,184,114,183,114,184,115,186,115,0,0
450 DATA 140,12,140,28,144,28,144,24,148,28,152,28,148,20,152,16,152,12,144,18,1
44,12,140,12,0,0
460 DATA 156,12,156,28,160,28,160,24,164,24,164,28,168,28,168,12,156,12,0,0,160,
16,160,20,164,20,164,16,160,16,0,0
470 DATA 172,12,172,28,176,28,176,28,176,18,184,28,188,28,188,12,184,12,184,20,1
76,12,172,12,0,0
480 DATA 192,12,192,28,196,28,196,24,200,28,204,28,200,20,204,16,204,12,196,18,1
96,12,192,12,0,0
490 DATA 208,12,208,28,212,28,212,24,216,24,216,28,220,28,220,12,208,12,0,0,212,
16,212,20,216,20,216,16,212,16,0,0
500 DATA 144,36,144,48,156,48,156,52,144,52,144,56,160,56,160,44,148,44,148,40,1
60,40,160,36,144,36,0,0
510 DATA 164,36,164,56,180,56,180,36,164,36,0,0,168,40,168,52,176,52,176,40,168,
40,0,0
520 DATA 184,36,184,56,188,56,188,48,196,48,196,44,188,44,188,40,200,40,200,36,1
84,36,0,0
530 DATA 204,36,204,40,212,40,212,56,216,56,216,40,224,40,224,36,204,36,-1,-1
540 PAINT(141,13),1:PAINT(157,13),1:PAINT(173,13),1:PAINT(193,13),1:PAINT(209,13
),1
550 PAINT(145,37),1:PAINT(165,37),1:PAINT(185,37),1:PAINT(205,37),1
560 FORT=1TO500:NEXTT:GOTO570
570 CURSOR15,180:PRINT"INSTRU Y or N"
580 AN$=INKEY$:IFAN$=""THEN580
590 IF AN$="Y"THENBEEP:GOTO 1550
600 IF AN$="N"THENBEEP:GOTO 680
610 FORT=1TO100:NEXTT:GOTO570
620 REM ****************************
630 REM *      STAR WAR C 1985      *
640 REM *            BY             *
650 REM *      JAN JACOBSEN         *
660 REM *   ADELAIDE USER'S CLUB    *
670 REM ****************************
680 C=12:MAG1
690 GOSUB1470:GOSUB1100:GOSUB1290
700 A=0:XX=102:YY=80:X1=0:Y1=0:RN=0:G=0:SO=0:MU=16:SC=0:MI=0
710 Y1=0:X1=INT(RND(1)*100)+50
720 RN=INT(RND(1)*15)+1
730 SPRITE0,(XX,YY),0,4
740 ONRNGOTO750,760,750,750,760,760,750,760,750,760,750,760,750,750,770
750 SPRITE1,(X1,Y1),4,15:G=1:SO=6:GOTO 780
760 SPRITE1,(X1,Y1),7,8:G=2:SO=7:GOTO 780
770 SPRITE1,(X1,Y1),12,10:G=3:SO=10
780 IN$=INKEY$:IFIN$=""THEN860
790 IN=ASC(IN$)
800 IFIN$=" "THENSOUND5,1,15:GOTO 930:GOTO 780
810 IFIN=29ANDXX>17THENXX=XX-20
820 IFIN=30ANDYY>15THENYY=YY-20
830 IFIN=31ANDYY<145THENYY=YY+20
840 IFIN=28AND XX<172THENXX=XX+20
850 SPRITE0,(XX,YY),0,4
860 X2=INT(RND(1)*3)
870 IFX2=0THENX1=X1+SO
880 IFX2=1THENX1=X1-SO
890 Y1=Y1+SO
900 IFMI>=10THEN990
910 IFX1>=182ORX1<=160RY1>=160THENMI=MI+1:BLINE(40,180)-(70,190),1,BF:CURSOR 45,
182:COLOR11:PRINTMI:GOTO 710
920 ONGOTO750,760,770
930 IF XX+8>=X1+4 AND XX+8<=X1+13 ANDYY+8>=Y1+4 AND YY+8<=Y1+13 THEN SPRITE1,(X1
,Y1),16,6:GOTO 950
940 SOUND5,1,0:GOTO810
950 SOUND4,0,15:FORA=0TO80:NEXT:SOUND0:IFSO=6THENSC=SC+30:GOTO980
960 IFSO=7THENSC=SC+80:GOTO980
970 IFSO=10THENSC=SC+200:GOTO980
980 BLINE(175,165)-(195,191),1,BF:CURSOR175,180:COLOR10:PRINTSC:GOTO710
990 CLS:SOUND0:CURSOR90,95:PRINT"YOUR SCORE";SC:CURSOR95,70:COLOR15:PRINT"GAME O
VER !"
1000 LINE(20,10)-(240,35),6,B
1010 CURSOR36,20:COLOR5:PRINTCHR$(17);"S T A R  W A R S  "
```

Sega

```
1020 PRINTCHR$(16)
1030 LINE(10,170)-(250,190),6,B
1040 COLOR14:CURSOR20,180:PRINT "KANKA Software (C)1985 By Jan Jacobsen"
1050 CURSOR52,130:PRINT"Do you want another game":COLOR7:CURSOR114,145:PRINT"(Y/
N)"
1060 IFINKEY$="Y"THENGOSUB 1290:GOTO700
1070 IFINKEY$="N"THEN1640
1080 GOTO1060
1090 RESTORE1480
1100 SCREEN2,2:COLOR,1,,1:CLS
1110 LINE(20,10)-(240,35),6,B
1120 LINE(10,170)-(250,190),6,B
1130 CURSOR36,20:COLOR5:PRINTCHR$(17);"S T A R  W A R S  "
1140 PRINTCHR$(16)
1150 SPRITE0,(50,70),4,15
1160 SPRITE1,(50,100),7,8
1170 SPRITE2,(50,130),12,10
1180 CURSOR80,78:COLOR15:PRINT"S C O U T     e 30 PTS"
1190 CURSOR80,108:COLOR7:PRINT"P A T R O L   e 80 PTS"
1200 CURSOR80,138:COLOR10:PRINT"F I G H T E R e 200 PTS"
1210 COLOR14:CURSOR20,180:PRINT "KANKA Software (C)1985 By Jan Jacobsen"
1220 RESTORE1260:FORJ=1TO110:READF:SOUND1,F,14:SOUND2,F+1,14:SOUND3,F*2,14:FORDE
=1TO30:NEXTDE,J
1230 READD
1240 IFD=0THENSOUND0:CLS:RETURN
1250 SOUND0,RETURN
1260 DATA 196,196,196,131,392,392,175,165,147,262,392,392,392,349,330,294,262,39
2,392,392,349,330,349,330,349,294,294,196,196,196,294,294,196,220,220,349,330,29
4,262,262,294,330,294,220,247,196,196,220,220,349,330,294,262,392,294,294,392,34
2,440,440,698
1270 DATA 698,659,587,523,523,587,659,587,440,494,392,392,523,466,415,392,349,31
1,294,262,392.784,196,196,196,262,392.392,349,330,294,523,392,392,392,349,330,29
4,523,392,392,392,349,330,349,294,294,294,294,0
1280 RETURN
1290 SCREEN2,2:COLOR,1:CLS
1300 FORP=0TO150:X=INT(RND(1)*200):Y=INT(RND(1)*170):E=INT(RND(1)*14)+2:PSET(X,Y
),E:NEXT
1310 CURSOR140,180:COLOR10:PRINT"SCORE":CURSOR10,180:COLOR10:PRINT"MISS"
1320 COLOR6:CURSOR215,5:PRINTCHR$(17);" S "
1330 COLOR15:CURSOR215,25:PRINT" T "
1340 COLOR4:CURSOR215,45:PRINT" A "
1350 COLOR7:CURSOR215,65:PRINT" R "
1360 COLOR14:CURSOR215,105:PRINT" W "
1370 COLOR10:CURSOR215,125:PRINT" A "
1380 COLOR6:CURSOR215,145:PRINT" R "
1390 PRINTCHR$(16)
1400 FORCI=1TO20STEP1
1410 CIRCLE(25,25),CI,11
1420 NEXTCI
1430 LINE(215,0)-(245,155),4,B
1440 LINE(210,0)-(250,155),4,B
1450 PAINT(211,1),4:PAINT(246,1),4
1460 RETURN
1470 RESTORE1500
1480 FORA=0TO19:READRE$:PATTERN$#A,RE$:NEXT:RETURN
1490 GOTO1480
1500 DATAF0F0C0C100010015,15000100C1C0F0F0,0F0F0383008000A8,A800800083030F0F
1510 DATA80C0E0A1E3A3E79E,9FE7A1E0A0E0C080,01030785C7C5E779,F9E7850705070301
1520 DATA0001021E3A57BAE7,0100000000000000,008040785CEA5DE7,8000000000000000
1530 DATA0000000000205089,47A77FA344990403,0000000000040A91,E2E5FEC5229920C0
1540 DATA43A3081C1E3F3F7F,FF7F7F3F3F1D3800,08183A70F2E1F8FE,FFFEF8ECC6829008
1550 SCREEN 1,1:CLS:PRINT "STAR WARS"
1560 PRINT"You are a member of the'FIGHTER SQUAD'
1570 PRINT "You are in your Fighter reday to blow up the EMPIRE Fighter"
1580 PRINT "The Fighter move randomly in a down   direction and are petty hard t
o catch.
1590 PRINT "IF you let more than ten EMPIRE       Fighter get away,you'll lose y
our job.
1600 PRINT "So hurry up!Jump into your space ship using the cursor key movements
 (up,   down,left and right),and speed to it!
1610 FORT=1TO 1500:NEXTT:PRINT :PRINT "Press any key to continue"
1620 IN$=INKEY$:IFIN$=""THEN1620
1630 GOTO 680
1640 SCREEN 1,1:CLS:PRINT "BYE":STOP
```

Apple Macintosh 128 Kbyte and 512 Kbyte

FREQUENCY HISTOGRAM

This business/education program plots a frequency histogram of a set of data after calculating the maximum and minimum data elements and determining the appropriate 'rounded' values and a suitable class interval.

It gives the user the option of saving the histogram to a Macpaint document, which may then be enhanced using Macpaint tools before being pasted into a Macwrite document. You can run an immediate printout of the histogram by pressing SHIFT CLOVER 4 (note the Macpascal drawing window can't be saved directly to a Macpaint document using SHIFT CLOVER 3).

Philip Cookson,
Northcote, Vic

```
program Frequency_Histogram (input, output);

{ Author : Mr. Philip Cookson                    Date : 10th September, 1985 }
{ This program reads a set of data and produces a frequency histogram  }
{ of the data. It automatically calculates 'neat' class intervals for the }
{ histogram and sorts the data into the appropriate class intervals. The }
{ histogram is then displayed in the drawing window.                    }

 const
  maxarraylength = 200; {maximum length of the data array}
  maxnclass = 11; {the maximum number of class intervals that can be generated}

 type
  data = array[1..maxarraylength] of real;
  freq = array[1..maxnclass] of real;

 var
  i : integer;
  xdata : data;
  xfreq : freq;
  xmin, xmax, classwidth : real;
  nclass, ndata : integer;
  response : char;
  title, filetitle : string;
  SaveTextRect : Rect;

 function log10 (x : real) : real; { Evaluates the logarithm to base 10 of x }
 begin {log10}
  if x < 0 then
   writeln('Invalid Argument')
  else
   log10 := ln(x) / ln(10)
 end; {log10}

 procedure initTextWindow;  { Initialize Text Window }
  var
   TextRect : Rect;
 begin {initTextWindow}
  SetRect(TextRect, 0, 20, 525, 340);
  SetTextRect(TextRect);
  ShowText
 end; {initTextWindow}
 procedure ReadTitle (var title : string);
{Reads Histogram Title from the input file}
 begin {ReadTitle}
  write('Enter a title for the histogram : ');
  readln(title);
  writeln;
 end; {ReadTitle}
```

Apple Macintosh 128 Kbyte and 512 Kbyte

```
procedure ReadData (var x : data;
          var n : integer);
{Reads data from the input file interactively. Note the use of the boolean }
{variables and the input pointer, to overcome Pascal's interactive input    }
{limitations. }
  var
    lineread, fileread : boolean;
begin {ReadData}
  n := 0;
  writeln('Enter the data (Enter  Q  to terminate data entry)');
  fileread := false;
  repeat
    lineread := false;
    repeat
      write('x[', n + 1 : 1, '] = ');
      if eoln then
        lineread := true
      else if (input^ = 'Q') or (input^ = 'q') then
        begin
          fileread := true;
          lineread := true
        end
      else
        begin
          n := n + 1;
          readln(x[n])
        end
    until lineread or (n = maxarraylength)
  until fileread;
  readln;
  writeln
end; {ReadData}
procedure FindMaxandMin (var x : data;
          n : integer;
          var xmax, xmin : real);
{ Determines the maximum and minimum of the input data array }
  var
    i : integer;
begin {FindMaxandMin}
  xmin := x[1];
  xmax := x[1];
  for i := 2 to n do
    begin
      if x[i] > xmax then
        xmax := x[i];
      if x[i] < xmin then
        xmin := x[i]
    end;
end; {FindMaxandMin}
```

Apple Macintosh 128 Kbyte and 512 Kbyte

```
procedure DetermineClassIntervals (var xmax, xmin : real;
          var nclass : integer;
          var classwidth : real);
{ An algorithm to determine 'neat' class intervals for the frequency }
{ histogram.  It returns neatly rounded values for 'xmin' and 'xmax',  }
{ and computes appropriate values for 'nclass' and 'classwidth'.       }
 const
  epsilon = 1.0e-7;
 var
  newxmin, newxmax : real;
  range : real;
  i, iclass, jclass, iorder : integer;
 function power (x : real;
           a : integer) : real;
{ determines the value of ' x raised to the power a ' }
{ for integer values fo a                              }
  var
   i : integer;
   product : real;
 begin {power}
  if x = 0 then
   power := 0
  else
   begin
    product := 1;
    if a > 0 then
     for i := 1 to a do
      product := product * x
    else if a < 0 then
     for i := 1 to a do
      product := product / x;
    power := product
   end
 end; {power}

 function convert (i : integer) : integer;
{ Used to determine the appropriate number of class intervals }
 begin {convert}
  case i of
   1 :
    convert := 1;
   10 :
    convert := 10;
   2, 5 :
    convert := 5;
   3, 6 :
    convert := 6;
   4, 8 :
    convert := 8;
```

Apple Macintosh 128 K*byte and* 512 K*byte*

```
    7:
     convert := 7;
    9:
     convert := 9
   end
end; {convert}
begin {DetermineClassIntervals}
 range := xmax - xmin;
 if (range <> 0) then
  begin
   iorder := round(log10(range));
   iclass := round(+0.5 - epsilon + range / power(10, iorder));
   jclass := convert(iclass);
   classwidth := iclass * power(10, iorder) / jclass;
   while range <= classwidth do
    classwidth := classwidth / 10;
   newxmin := round(-0.5 + epsilon + xmin / classwidth) * classwidth;
   newxmax := round(+0.5 - epsilon + xmax / classwidth) * classwidth;
   if xmax = newxmax then
    newxmax := newxmax + classwidth;
   while xmin >= (newxmin + classwidth - epsilon) do
    newxmin := newxmin + classwidth;
   while xmax < (newxmax - classwidth) do
    newxmax := newxmax - classwidth;
   xmax := newxmax;
   xmin := newxmin;
   nclass := round((xmax - xmin) / classwidth)
  end
 else
  begin
   nclass := 1;
   classwidth := epsilon
  end
end; {DetermineClassIntervals}

procedure SortintoClassintervals (var x : data;
          ndata : integer;
          var xfreq : freq;
          xmax, xmin : real;
          classwidth : real);
{ Sorts the data into the appropriate class intervals. Note that the }
{ data does not have to be ordered in any way. }
 var
  i, k : integer;
begin {SortintoClassintervals}
 for i := 1 to maxnclass do
  xfreq[i] := 0;
```

```
 for i := 1 to ndata do
  begin
   k := 1 + trunc(abs(x[i] - xmin) / classwidth);
   xfreq[k] := xfreq[k] + 1
  end
end; {SortintoClassIntervals}

procedure PercentageFrequency (var xfreq : freq;
          ndata : integer);
 var
  i : integer;
begin {PercentageFrequency}
 for i := 1 to maxnclass do
  xfreq[i] := xfreq[i] * 100 / ndata;
end; {PercentageFrequency}

procedure InitDrawingWindow;
 { Initialize Drawing Window }
 var
  DrawRect : Rect;
begin {InitDrawingWindow}
 SetRect(DrawRect, 0, 0, 532, 362);
 SetDrawingRect(DrawRect);
 ShowDrawing
end; {InitDrawingWindow}

procedure DrawHistogram (frequency : freq;
          nclasses : integer;
          mindata, classwidth : real;
          title : string);
 const

  Ftop = 60;              { coordinates for Histogram Frame }
  Fbottom = 300;
  Fleft = 35;
  Fright = 475;

 var
  Frame, Bar : Rect;
  i, width, Bleft, spacing, ndp : integer;

  function iscale (x : real) : integer;
  begin {iscale}
   iscale := Fbottom - round((Fbottom - Ftop) * x / 100)
  end; {iscale}
 begin {DrawHistogram}
  InitDrawingWindow;
  TextSize(12);
```

NOTES:

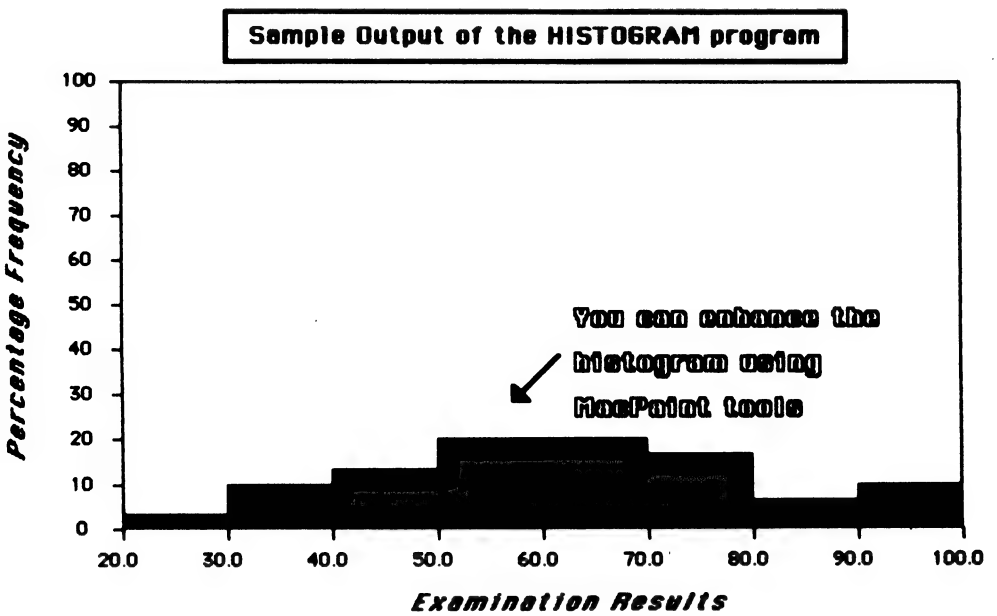
Apple Macintosh 128 Kbyte and 512 Kbyte

```
  TextFace([bold]);
  MoveTo(Fleft + round(((Fright - Fleft) - 8 * length(title)) / 2), Ftop - 20);
  DrawString(title);
  TextSize(9);
  TextFace([]);
  width := round((Fright - Fleft) / nclasses);
  SetRect(Frame, Fleft, Ftop, Fright, Fbottom);
  for i := 1 to nclasses do
   begin
    Bleft := Fleft + (i - 1) * width;
    SetRect(Bar, Bleft, iscale(frequency[i]), Bleft + width, Fbottom);
    FillRect(Bar, gray);
    MoveTo(Bleft, Fbottom);
    LineTo(Bleft, Fbottom + 5);
    FrameRect(Bar);
   end;
  MoveTo(Fright, Fbottom);
  LineTo(Fright, Fbottom + 5);
  FrameRect(Frame);
  for i := 0 to 10 do
   begin {Label Vertical Axis}
    spacing := round(i * 10 * (Fbottom - Ftop) / 100);
    MoveTo(Fleft - 30, Fbottom + 3 - spacing);
    WriteDraw(i * 10 : 3);
    MoveTo(Fleft - 5, Fbottom - spacing);
    LineTo(Fleft, Fbottom - spacing);
   end;
  ndp := round(-log10(classwidth));
  if ndp > 8 then
   ndp := 8;
  if ndp <= 0 then
   ndp := 1;
  for i := 0 to nclasses do
   begin {Label Horizontal Axis}
    MoveTo(Fleft - 25 + width * i, Fbottom + 20);
    WriteDraw(mindata + i * classwidth : 8 : ndp);
   end;
 end; {DrawHistogram}

begin {Main program body}
 GetTextRect(SaveTextRect);
 InitTextWindow;
 ReadTitle(title);
 ReadData(xdata, ndata);
 FindMaxandMin(xdata, ndata, xmax, xmin);
 DetermineClassIntervals(xmax, xmin, nclass, classwidth);
 SortintoClassIntervals(xdata, ndata, xfreq, xmax, xmin, classwidth);
 PercentageFrequency(xfreq, ndata);
 write('Do you wish to save the Histogram to a MacPaint file [Y/N]? ');
```

Apple Macintosh 128 K*byte and* 512 K*byte*

```
  read(response);
  if (response = 'Y') or (response = 'y') then
    filetitle := NewFileName('Save drawing to :');
  InitDrawingWindow;
  DrawHistogram(xfreq, nclass, xmin, classwidth, title);
  if (response = 'Y') or (response = 'y') then
    SaveDrawing(filetitle);
  SetTextRect(SaveTextRect);
end. {Main program body}
```



NOTES:

VZ200

YAHTZEE

This is a VZ200 version for the dice game Yahtzee, designed for an unlimited number of players.

Each player throws his or her dice up to three times each turn. After the first and second throws you can hold any dice you wish to keep, re-throwing the balance. After the third throw you must enter your score in the table provided.

Once a score has been recorded for a particular category, that category can't be used again. The game ends after 13 rounds.

Because of the limitations of the printer used to produce the listing it's wise to include the graphics [shift J's] in lines 2020 and 2050. The sections underlined should be inserted in inverse text.

The program occupies 5.9 Kbytes of memory.

Ian Thompson,
Collaroy Plateau, NSW

Main Variable Used

N$( )	Player's name
S1$( )	Titles of category
S2$( )	Description of category
SC$( )	Update score for each category
DF( )	Spots on dice
SC( )	Score
ND( )	Random number for dice
NP	Number of players
IP	User update of scoresheet
TURN	Turn number

Main Routines

0- 46	Title graphics
100	Initialises screen background Clears memory for variables
130	Input players' names
140	Initialises variables
160-170	Input players names
165	Limits player's name to 11 characters
180-190	Set number of player turns per number of players
210	Random number generator for dice throw
290-320	Print score table
330-420	User update of score
470-570	Updates total score
600-650	Displays final score and placings
1000-1120	Data statements for score table
1130-1160	Data statements for spots on dice
2020-2050	Displays dice
9000-20040	User update of score subroutine
21000-22140	Instructions

```
0 '****************************
1 '*  VZ-200  Y A H T Z E E   *
2 '* IAN THOMPSON - COLLAROY  *
3 '****************************
4 CLS:SOUND 25,6:COLOR,0
5 FOR X=1 TO 32:POKE 28671+X,204:POKE 29151+
X,195
6 POKE 28672,174:POKE 28703,173:POKE 29152,1
71:POKE 29183,167
7 NEXT X
8 FOR N=28704 TO 29120 STEP 32
9 POKE N,202
10 NEXT N
11 FOR O=28735 TO 29151 STEP 32
12 POKE O,197
13 NEXT O
22 PRINT@106," YAHTZEE "
24 A$=" IAN A.THOMPSON "
26 B$="COLLAROY PLATEAU"
28 FOR N=1 TO LEN(A$)
30 PRINT@231,RIGHT$(A$,N);
32 PRINT@263,RIGHT$(B$,N);
34 NEXT
35 FOR I=1 TO 500:NEXTI
36 PRINT@454,"INSTRUCTIONS (Y/N)?"
38 IF INKEY$<>"" THEN 38
40 A$=INKEY$
42 IF A$="N" THEN SOUND30,1:GOTO100
44 IF A$<>"Y" THEN 40
45 SOUND30,1
46 GOSUB 21000:'INSTRUCTIONS
100 POKE 30744,0:COLOR 5,0:CLEAR 1000
120 R=RND(0)
130 CLS:PRINT@128,"NO. OF PLAYERS";:INPUT NP
135 SOUND 31,1
140 DIM SC$(13,NP),S1$(13),S2$(13),N$(NP),DF
(6,6),SC(NP),YF(NP)
145 GOSUB 1000
150 FOR I=1 TO NP
160 CLS:PRINT@128,"PLAYER #";I;:INPUT"'S NAM
E";N$(I)
162 SOUND 31,1
165 IF LEN(N$(I))>11 THEN SOUND 20,1:10,1:GO
TO 160
170 NEXT
180 FOR TURN = 1 TO 13
190 FOR PL=1 TO NP
210 FOR R=1 TO 5:ND(R)=RND(6):NEXT
220 GOSUB 2000
230 GOSUB 3000
240 PRINT@416,"REMEMBER THESE,THEN"
250 PRINT"HIT ANY KEY TO CONTINUE":A$=INKEY$
260 A$=INKEY$:IF A$="" THEN 260
270 FOR I=1 TO 6:N(I)=0:NEXT
280 FOR I=1 TO 5:N(ND(I))=N(ND(I))+1:NEXT
290 CLS:PRINT"CHOOSE A CATEGORY,";N$(PL)
300 FOR I=1 TO 13:PRINTUSING"##) ";I;
310 PRINTS1$(I);S2$(I);SC$(I,PL)
320 NEXT
330 PRINT:INPUT"WHICH [1-13]";IP
340 IF IP<1 OR IP>13 THEN 290
345 IF IP=12 THEN 12000
350 IF SC$(IP,PL)<>"" THEN 15000
360 IF IP<7 THEN SC$(IP,PL)=STR$(IP*N(IP))
370 IF IP=7 OR IP=8 THEN 7000
380 IF IP=9 THEN 9000
390 IF IP=10 THEN 10000
400 IF IP=11 THEN 11000
420 IF IP=13 THEN 13000
```

VZ200

```
430 NEXT PL:NEXT TURN
440 FOR PL=1 TO NP:FOR I=1 TO 13
460 NEXT
470 FOR I=1 TO 6
480 SC(PL)=SC(PL)+VAL(SC$(I,PL)):NEXT
490 IF SC(PL)>62 THEN SC(PL)=SC(PL)+35
500 FOR I=7 TO 13
510 SC(PL)=SC(PL)+VAL(SC$(I,PL))
520 NEXT:NEXT
530 FOR I=1 TO NP-1
540 HI=0:FOR J=1 TO NP
550 IF SC(J)>HI THEN HI=SC(J):P=J
560 NEXT
570 D=SC(I):SC(I)=SC(P):SC(P)=D
580 D$=N$(I):N$(I)=N$(P):N$(P)=D$
585 NEXT
590 SOUND 20,1:SOUND 10,1:SOUND 20,1
600 CLS:PRINT"AND THE PLACINGS ARE";
620 PRINT:PRINT:FOR I=1 TO NP
630 PRINTUSING"##";I;
640 PRINT"] ";N$(I);TAB(25);SC(I)
650 NEXT
660 PRINT@480,"ANOTHER GAME (Y/N)?";
670 GOSUB 20000
680 IF YN$="Y" THEN RUN
690 CLS:PRINT@162,"THANKS FOR THE GAME  BYE
":END
1000 'DATA STATEMENTS
1040 FOR I=1 TO 13
1050 READ S1$(I),S2$(I):NEXT
1060 DATA"ACES",".  [SUM OF 1'S] -","TWOS","
.  [SUM OF 2'S] -"
1070 DATA"THREES",".[SUM OF 3'S] -","FOURS",
". [SUM OF 4'S] -"
1080 DATA"FIVES",". [SUM OF 5'S] -","SIXES",
". [SUM OF 6'S] -"
1090 DATA"3 OF A KIND",".  [SUM] -","4 OF A
KIND",".  [SUM] -"
1100 DATA"FULL HOUSE",".    [25] -","SM. STR
AIGHT",".  [30] -"
1110 DATA"LGE. STRAIGHT",". [40] -","YAHTZEE
",".       [50] -"
1120 DATA"CHANCE",".        [SUM] -"
1130 FOR I=1 TO 6:FOR J=1 TO I
1140 READ DF(I,J):NEXT:NEXT
1150 DATA 66,33,99,1,66,131,33,35,97,99
1160 DATA 1,3,66,129,131,1,3,65,67,129,131
1190 RETURN
2000 CLS:PRINT N$(PL);"'S ROLL"
2010 FOR R=96 TO 224 STEP 32:FOR S=2 TO 26 S
TEP 6
2015 COLOR 5
2020 PRINT@R+S,"   ";'THREE SHIFT J'S
2030 NEXT:NEXT
2040 FOR D=1 TO 5:FOR N=1 TO ND(D)
2045 COLOR 3
2050 PRINT@91+D*6+DF(ND(D),N)," ";'ONE SHIFT
 J
2060 NEXT:NEXT
2070 RETURN
3000 FOR K=1 TO 2:F=1
3010 FOR J=1 TO 5
3020 P=252+J*6:RR(J)=0
3030 PRINT@P,"^^^"
3035 PRINT:PRINT
3040 PRINT"REROLL THIS ONE [Y/N]?"
3050 PRINT"[S FOR SCOREBOARD]":PRINT"[M FOR
MISTAKE]":YN$=INKEY$
3060 YN$=INKEY$
3070 IF YN$<>"Y" AND YN$<>"N" AND YN$<>"S" A
ND YN$<>"M" THEN3060
3072 IF YN$="Y" THEN SOUND 20,1
3074 IF YN$="N" THEN SOUND 10,1
3076 IF YN$="S" THEN SOUND 15,1
3078 IF YN$="M" THEN SOUND 20,2;10,1
3080 IF YN$<>"S" THEN 3130
3090 CLS:PRINT TAB(5);N$(PL);"'S SCORES"
3100 FORI=1TO13:PRINTUSING"##] ";I;
3105 PRINT S1$(I);S2$(I);SC$(I,PL)
3110 NEXT:PRINT"HIT ANY KEY TO RETURN:":A$=I
NKEY$
3120 A$=INKEY$:IF A$="" THEN 3120 ELSE GOSUB
 2000:GOTO 3020
3130 PRINT@P,"   "
3140 IF YN$="M" THEN 3010
3150 IF YN$="Y" THEN RR(J)=1
3160 NEXT
3170 FOR I=1 TO 5:IF RR(I)=1 THEN ND(I)=RND(
6):F=0
3180 NEXT:IF F THEN K=2
3190 GOSUB 2000
3200 NEXT
3210 RETURN
7000 FOR I=1 TO 6:IFN(I)>IP-5 THEN 7030
7010 NEXT
7020 GOTO 16000
7030 SC=0
7040 FOR I=1 TO 5:SC=SC+ND(I):NEXT
7050 SC$(IP,PL)=STR$(SC)
7060 GOTO 430
9000 FOR I=1 TO 6
9010 IF N(I)>2 THEN N(I)=N(I)-3:GOTO 9040
9020 NEXT
9030 GOTO 16000
9040 FOR I=1 TO 6
9050 IF N(I)>1 THEN 9080
9060 NEXT
9070 GOTO 16000
9080 SC$(9,PL)=" 25"
9090 GOTO 430
9090 GOTO 430
10000 FOR I=1 TO 3:F=1:FOR J=I TO I+3
10010 IF N(J)=0 THEN F=0
10020 NEXT
10030 IF F THEN 10060
10040 NEXT
10050 GOTO 16000
10060 SC$(10,PL)=" 30"
10070 GOTO 430
11000 FOR I=1 TO 2:F=1:FOR J=I TO I+4
11010 IF N(J)=0 THEN F=0
11020 NEXT
11030 IF F THEN 11060
11040 NEXT
11050 GOTO 16000
11060 SC$(11,PL)=" 40"
11070 GOTO 430
12000 FOR I=1 TO 6
12010 IF N(I)=5 THEN 12040
12020 NEXT
12030 GOTO 16000
12040 SC$(12,PL)=" 50"
12050 IF YF(PL) THEN SC$(12,PL)=STR$(VAL(SC$
(12,PL))+100)
12060 YF(PL)=1
12070 GOTO 430
13000 SC=0:FOR I=1 TO 5
13010 SC=SC+ND(I):NEXT
13020 SC$(13,PL)=STR$(SC)
13030 GOTO 430
```

VZ200

```
15000 SOUND 15,1:CLS
15010 PRINT@128,"YOU'VE ALREADY DONE"
15020 PRINT"THE ";S1$(IP);" ";N$(PL)
15030 FOR I=1 TO 2000:NEXT
15040 GOTO 290
16000 SOUND 15,1:CLS
16010 PRINT@128,"YOU'RE NOT ELIGIBLE FOR"
16020 PRINT"A ";S1$(IP);" ";N$(PL)
16025 IF IP=12 AND YF(PL) THEN SOUND 0,8:GOT
O 290
16030 PRINT:PRINT:PRINT"DO YOU WANT IT ANYWA
Y [Y/N]?";
16040 GOSUB 20000
16050 IF YN$="N" THEN 290
16060 SC$(IP,PL)=" 0"
16070 IF IP=12 THEN YF(PL)=1
16080 GOTO 430
20000 YN$=INKEY$
20010 YN$=INKEY$:IF YN$="" THEN  20010
20020 IF YN$<>"Y" AND YN$<>"N" THEN 20000
20030 IF YN$="Y" THEN SOUND 20,1 ELSE SOUND
10,1
20040 RETURN
21000 CLS:PRINT"INSTRUCTIONS"
21010 PRINT:PRINT"IN THIS DICE GAME EACH PLA
YER"
21020 PRINT"CAN THROW UP TO THREE TIMES EACH
";
21030 PRINT"TURN. AFTER THE FIRST THROW, HE"
21040 PRINT"CAN SET ASIDE ANY DICE HE WISHES
";
21050 PRINT"TO KEEP,AND RETHROW THE BALANCE.
";
21060 PRINT"HE CAN DO THE SAME AFTER THE"
21070 PRINT"SECOND AND THIRD THROWS. HE CAN,
";
21080 PRINT"OF COURSE, STOP BEFORE THE THIRD
";
21090 PRINT"THROW IF HE WISHES."
21100 PRINT"ONCE THE PLAYER HAS DECIDED TO"
21110 PRINT"STOP, HE MUST DECIDE INTO WHICH"
21120 PRINT"CATEGORY TO ENTER HIS SCORE."
21130 GOSUB 22100
21140 DIM S1$(13),S2$(13)
21150 FOR I=1 TO 13
21152 READ S1$(I),S2$(I):NEXT
21154 CLS
21156 PRINT
21160 FOR I=1 TO 13
21170 PRINT S1$(I);S2$(I):NEXT
21172 PRINT@300,"[SUM OF HOUSE] -"
21174 PRINT@334,"[1,2,3,4,5,] -"
21176 PRINT@366,"[2,3,4,5,6,] -"
21178 PRINT@394,"[FIVE OF A KIND] -"
21180 PRINT@427,"[ANY FIVE DICE] -"
21200 GOSUB 22100
21210 CLS:PRINT"THE GAME ENDS AFTER 12 ROUND
S."
21220 PRINT"ONCE A SCORE HAS BEEN RECORDED"
21230 PRINT"FOR A PARTICULAR CATEGORY, THAT"
21240 PRINT"CATEGORY CAN'T BE USED AGAIN."
21250 GOSUB 22100
21260 RETURN
22100 PRINT@485,"PRESS <C> TO CONTINUE";
22110 IF INKEY$<>"C" THEN 22100
22120 IF INKEY$="" THEN 22100
22130 IF INKEY$="C" THEN 22130
22140 SOUND 30,1:RETURN
```

Most BASICs

SORTING OUT THE SORTS

The program listed with this article was developed to test the speed and efficiency of four sorting algorithms: Insertion, Shell, Quick and Selection. The program will probably run as is in most BASICs.

What's usually required of a sort is to put a list of names into alphabetical order, but the average textbook seems to present sorts for numbers with no indication of the best choice for a particular task.

The choice of sorting algorithms is broad: more than three dozen are known, spread across some hundred texts. The most popularly presented, and the slowest if list size is more than 11, is the bubble sort.

Fortunately, the choice can be narrowed down to short algorithms which work in RAM only and are not bubbly sorts in disguise!

The program

The first program line, line 30, asks for the size of the array to be generated and sorted. Start with a choice of 10 to check the sorts are working as expected.

Lines 70-130 generate the required number of capital letters and place them in the array. The variable CD is set at 64, one less than the ASCII code for 'A'. A number is generated in line 90 and added to CD. If this number is acceptable, the character it represents is placed in array CH$. If not, C is decremented by one and the process is repeated. The loop runs until BN (big number!) letters are placed in the array. Line 130 prints out the unsorted list and could be omitted from the program.

The straight Insertion sort and the Shell (insertion algorithm) sort are particularly useful.

The Selection sort is always the slowest, since the same number of compares and swaps is made if the list is random, or if only one element is out of order. The Quick sort isn't much better if only a few

Most BASICs

elements are out of order. If the list is random with more than 500 elements Quick sort is useful, but only if there is no shortage of RAM to store the two extra arrays the sort requires.

The Shell sort is best for a random list, and the Insertion sort is best when only a few items in the list are out of order. For most work the Insertion sort will do. For instance, a mailing list is only truly random when first typed in, and thereafter only insertions need to be sorted. The Insertion sort is simplest to understand: search forward for an out-of-order element, then search back through the list and insert the element in its proper place. The Shell sort is a little more complicated: elements compared are a specified distance apart, which decreases until only adjacent elements are compared.

Mr Jankowski,
Timaru, New Zealand

```
10 REM SORT MEASURE by LZ Jankowski
15 REM COPYRIGHT July 1985
20 :
30 CLS: INPUT "# of items in list "; BN: CLS
40 DIM CH$(BN), A(BN), B(BN)
50 :
60 REM ---------Create random list of letters-----------------
70 CLS: PRINT TAB( 20)"* Programming *": CO=64
80 FOR C=1 TO BN: CO=CO+10: IF CO=94 THEN CO=64
90 :  X=INT(10*RND(1))+CO
100 :    IF X<65 OR X>90 THEN C=C-1: GOTO 120
110 :     CH$(C)=CHR$(X)
120 NEXT : CLS: PRINT "Random list is": PRINT : PRINT
130 FOR C=1 TO BN: PRINT C"=" CH$(C) SPC( 4);: NEXT : PRINT : PRINT
140 :
150 REM -------------------MENU--------------------------------
160 PRINT : PRINT : PRINT : PRINT TAB( 15)"1> Insertion Sort"
170 PRINT TAB( 15)"2> Shell      Sort"
180 PRINT TAB( 15)"3> Quick      Sort": PRINT TAB(15)"4> Selection Sort
190 PRINT : INPUT "Choice "; C
200 IF C<1 OR C>4 THEN RUN
210 :
220 PRINT !(28): PRINT "* Sorting *": PRINT : PRINT
230 ON C GOSUB 500, 400, 570, 750: GOSUB 350
240 :
250 REM ------Now sort 'A' from bottom to top of list---------
260 FOR C=1 TO 40: PRINT "-";: NEXT
270 PRINT : PRINT "Sorting 'A' from bottom to top of list": PRINT
280 GOSUB 500: GOSUB 350: GOSUB 400: GOSUB 350
290 GOSUB 570: GOSUB 350: GOSUB 750: GOSUB 350
300 :
310 PRINT : INPUT "RUN again ";Q$: IF LEFT$(Q$,1)="Y" THEN RUN
320 END
330 :
340 REM -------Print sorted list & # of compares & swaps------
350 FOR C=1 TO BN: PRINT C "=" CH$(C) SPC( 4);: NEXT
360 PRINT : PRINT : PRINT "Compares= "CM,"Swaps= "S: PRINT
370 CH$(BN)="A": CM=0: S=0: RETURN
380 :
390 REM ------Shell sort based on Insertion algorithm---------
400 I=(2^INT(LOG(BN)/LOG(2)))-1
410 I=INT(I/2)
420 IF I<1 THEN 470
430 FOR N=1 TO I: FOR C=N+I TO BN STEP I: M=C: C$=CH$(M)
440 CM=CM+1: IF CH$(M-I)<=C$ THEN 460
450 CH$(M)=CH$(M-I): S=S+1: M=M-I: IF M>I THEN 440
460 CH$(M)=C$: NEXT C: NEXT N: GOTO 410
470 PRINT "SHELL SORT": RETURN
480 :
490 REM --------------Insertion Sort---------------------------
500 FOR N=2 TO BN: M=N: C$=CH$(M)
510 CM=CM+1: IF CH$(M-1)<=C$ THEN 530
520 S=S+1: CH$(M)=CH$(M-1): M=M-1: IF M>1 THEN 510
530 CH$(M)=C$: NEXT
540 PRINT "INSERTION SORT": RETURN
550 :
560 REM ----Quicksort best for very long, random lists--------
570 SP=1: A(1)=1: B(1)=BN
580 FI=A(SP): SI=B(SP): SP=SP-1
590 SF=FI: SS=SI: C$=CH$(INT((FI+SI)/2))
600 CM=CM+1: IF CH$(SF)>=C$ THEN 630
610 SF=SF+1
620 GOTO 600
630 CM=CM+1: IF C$>=CH$(SS) THEN 650
640 SS=SS-1: GOTO 630
650 IF SF>SS THEN 670
660 S=S+1: E$=CH$(SF): CH$(SF)=CH$(SS): CH$(SS)=E$: SF=SF+1: SS=SS-1
670 IF SF<=SS THEN 600
680 IF SF>=SI THEN 700
690 SP=SP+1: A(SP)=SF: B(SP)=SI
700 SI=SS: IF FI<SI THEN 590
710 IF SP>0 THEN 580
720 PRINT "QUICKSORT": RETURN
730 :
740 REM ----Selection Sort, dreadful! CM=(N-1)*N/2 Swaps=N-1--
750 FOR N=1 TO BN-1:M=N: FOR C=N+1 TO BN:CM=CM+1:IFCH$(M)>CH$(C)THENM=C
760 NEXT: S=S+1: C$=CH$(N): CH$(N)=CH$(M): CH$(M)=C$: NEXT
770 PRINT "SELECTION SORT": RETURN
```

Microsoft BASIC

SUPERMAIL — THE UNIVERSAL MAILING LIST MANAGER

NEED TO PRINT mailing labels, envelopes or personalised mail from a mailing list? Supermail can solve your problems. There are versions of Supermail, written in Microsoft BASIC, for the IBM PC, NEC PC 8201 and Tandy Model 100. With a few adaptations to line 261, 262 or 263, which control serial port syntax on the target machine, all machines using Microsoft BASIC should be okay.

Printers
Serial and parallel printers are supported, and the user chooses which to use as the program runs. This is achieved by avoiding the use of LPRINT, nominating a numbered file to be the serial or parallel port, and printing to that file. Lines 261, 262 and 263 call for serial port parameters of 300 baud, 8-bit word length, one stop bit and no parity. To use printers requiring other parameters for the serial port you'll have to modify line 261, 262 or 263.

Program Features
Supermail creates a Mailmerge-compatible database, and from it makes mailing labels, addresses envelopes and creates personalised bulk mail. Room is provided for additional routines to sort the database and to make reports.

Database Structure
The first field has a fixed length and is for the postcode. The remaining fields, starting with 'lastname', are of variable length for most efficient use of memory in the Tandy 100 and NEC PC 8201. These two fields allow easy sorting by postcode into areas, or sorting into alphabetical order according to surname. Only sequential files are used, due to the file limitations of the NEC and Tandy.

Program Structure
The program starts with initialisation routines and an opening menu. Separate subroutines are used to add names to the database, to print labels, envelopes and letters, and for error handling. As printed, the listing is for an IBM PC. To use it on a NEC PC 8201 put a REM at the start of line 261, and remove the ' from the start of line 262. For the Tandy 100, add a REM to line 261, and remove the ' from line 263.

File Buffers
Line 150 has the statement MAXFILES=3. This is required for the NEC and Tandy to allocate file buffers for input and output. It isn't required for the IBM but has no ill effect, since IBM PC BASIC allocates three buffers automatically and only sees the MAXFILES as an unused variable.

Optional Routines
On my NEC 8201, I took the sort routine from page 181 of the Tandy 100 manual, renumbered it, replaced the END statement with a CLOSE:GOTO 570, and put it between lines 5070 and 5995. IBM PC-DOS 2.00 (and later) has a SORT filter that does this job very efficiently, and the space could be better used for a report generation routine.

Remarks Fields
The 'Add a name' routine creates two remarks fields in the database. The first of these allows 255 characters of remarks. Line 1160 creates the second remarks field as an empty field. If more than 255 characters of remarks are required, replace line 1160 with a routine similar to the one in line 1150. None of the printing routines, as written, displays the remarks. A report routine could include them, and they could be used for diverse purposes, such as showing which club members are overdue with annual membership fees!

Form Letters
Form letters require preformatted text for the body of the letter, with a carriage return/linefeed at the end of each line. PC-Write will create such files for IBM. For the Tandy or NEC, manual splitting of text into lines with the return key will be required. Files created by programs such as Wordstar in document mode include all sorts of characters not in the normal ASCII sequence. Such files should be put through a stripper program to convert them to pure ASCII before using Supermail.

Supermail is easy to use and is fully menu driven. It is a simple program, but quite powerful and very useful. Even if you only use it to create a data file for Mailmerge, you'll quickly see its worth. □

```
10 ' ********************************
20 ' *                              *
30 ' *        ***superMAIL***       *
40 ' *                              *
50 ' ********************************
60 '
70 ' copyright John Hepworth
80 ' Haberfield, Australia, August 1984
90 '
100 ' Individual use permitted.
100 '
110 ' Not to be resold or used for
120 ' commercial purposes
130 '
140 '
150 CLEAR 1000:MAXFILES=3   'maxfiles is
     needed by NEC & TANDY laphelds
160 DIM NA$(3,8) 'NA$ is the name and
     address array
170 CLS:ER=10:'OF$="Printer chosen"
180 ON ERROR GOTO 3570
190 PRINT:PRINT:PRINT SPACE$(13);
     "***superMAIL***"
200 PRINT:PRINT:PRINT SPACE$(8);
     "multipurpose mass mailer"
210 PRINT:PRINT SPACE$(8); "Copyright
     John Hepworth"
220 FOR N=1 TO 1000:NEXT N
230 INPUT "<P>arallel or <S>erial
     printer";OS$
240 N=INSTR("PpSs",OS$)
250 IF N=0 THEN PRINT "Try again":GOTO
     230
261 IF N=1 OR N=2 THEN OPEN "LPT1" FOR
     OUTPUT AS#3 ELSE OPEN
     "COM1:3,n,8,1" FOR OUTPUT AS#3
     'IBM version of line 260
262 ' IF N=1 OR N=2 THEN OPEN "LPT:" FOR
     OUTPUT AS#3 ELSE OPEN "COM:3n81x"
     FOR OUTPUT AS#3    'NEC version -
     line 260
263 ' IF N=1 OR N=2 THEN OPEN "LPT:" FOR
     OUTPUT AS#3 ELSE OPEN "COM:38n1e"
     FOR OUTPUT AS#3    'TANDY version -
     line 260
500 '
510 ' ********************************
520 ' *                              *
```

Microsoft BASIC

```
53Ø ' *       ** first menu **         *
54Ø ' *                                *
55Ø ' **********************************
56Ø '
57Ø CLS:PRINT:PRINT"   Select one
     alternative"
58Ø PRINT"<A> Add a name and address to
     the file"
59Ø PRINT"<L> Print labels"
6ØØ PRINT"<E> Print envelopes"
61Ø PRINT"<B> Print bulk mail"
62Ø PRINT"<Q> Quit, and return to main
     menu"
625 PRINT"<O> Other option"
63Ø INPUT CH$
64Ø ON INT(INSTR("AaL1EeBbQqOo",CH$)/2+.5)
     GOTO 1Ø7Ø,157Ø,2Ø7Ø,257Ø,3Ø3Ø,5Ø3Ø
65Ø GOTO 57Ø

1ØØØ '
1Ø1Ø ' **********************************
1Ø2Ø ' *                                *
1Ø3Ø ' *  Adding a name to the file     *
1Ø4Ø ' *                                *
1Ø5Ø ' **********************************
1Ø6Ø '

1Ø7Ø FILES:INPUT "TO WHICH FILE DO YOU
     WANT TO ADD NAMES"; A4$
1Ø8Ø OPEN A4$ FOR APPEND AS #1
1Ø85 FOR N= Ø TO 8:A4$(N)="":NEXT N
1Ø9Ø INPUT "Courtesy title <eg Mrs, Ms,
     Mr, Dr >";A4$(1)
11ØØ INPUT "First name(s) or initials ";
     A4$(2)
111Ø INPUT "Last name ";A4$(3)
112Ø INPUT "Street number and street
     name";A4$(4)
113Ø INPUT "Town (or suburb) and
     state";A4$(5)
114Ø INPUT
     "Postcode";A4$(6):PC=LEN(A4$(6)):
     IF PC<4 THEN A4$(6)=SPACE$(4-PC)
     +A4$(6) ELSE IF PC>4 THEN PRINT
     "POSTCODE TOO LONG":GOTO 114Ø
115Ø INPUT "Remarks";A4$(7)
116Ø A4$(8)=""
117Ø FOR N=1 TO 8:
     CO=INSTR(A4$(N),","):IF CO>Ø THEN
     MID$(A4$(X),CO,1)=" ":NEXT N
118Ø PRINT #1,
     A4$(6)","A4$(3)","A4$(1)","A4$(2)",
     "A4$(4)","A4$(5)","A4$(7)","A4$(8)
119Ø INPUT "add another name and
     address"; IP$
12ØØ IF INSTR("YyNn",IP$)=Ø GOTO 119Ø
121Ø IF INSTR("Yy",IP$) THEN 1Ø9Ø
122Ø CLOSE #1:GOTO 57Ø

15ØØ '
151Ø ' **********************************
152Ø ' *                                *
153Ø ' *    label printing routine      *
154Ø ' *                                *
155Ø ' **********************************
156Ø '
157Ø CLS:PRINT:PRINT"printing labels"
158Ø INPUT "No of labels across sheet (1
     to 3)";LA:IF LA<1 OR LA>3 GOTO 158Ø
159Ø INPUT "No of lines per label (3 or
     more)";LL:IF LL<3 GOTO 159Ø
16ØØ INPUT "No columns of type across
     sheet";LS
161Ø CLS:FILES:INPUT "Filename for names
     and addresses";NA$:OF$=NA$:OPEN NA$
     FOR INPUT AS #1
162Ø INPUT "HOW MANY COPIES OF EACH
     LABEL";CL
163Ø CN=Ø
164Ø CN=CN+1:I$=INKEY$:IF I$=CHR$(27)
     THEN CLOSE #1:GOTO 57Ø 'hit
     <escape> to abort print run and
     return to end.
165Ø FOR N=1 TO LA
166Ø     FOR M=1 TO 8
167Ø       INPUT #1, NA$(N,M)
168Ø       IF (EOF(1) AND N<LA) THEN
            GOTO 172Ø
169Ø     NEXT M
17ØØ NEXT N
171Ø GOTO 177Ø
172Ø FOR X=N+1 TO LA
173Ø     FOR M=1 TO 8
174Ø       NA$(X,M)=""
175Ø     NEXT M
176Ø NEXT X
177Ø FOR X= 1 TO LA
178Ø    L$=NA$(X,3)+" "+NA$(X,4)+
        " "+NA$(X,2)
179Ø    PRINT L$
18ØØ    PRINT #3, SPACE$(5) +L$+
          SPACE$(LS\LA -5-LEN(L$));
181Ø NEXT X
182Ø PRINT #3,""
183Ø FOR X= 1 TO LA
184Ø    L$=NA$(X,5)
185Ø    PRINT #3, SPACE$(5) +L$+
          SPACE$(LS\LA -5-LEN(L$));
186Ø NEXT X
187Ø PRINT #3,""
188Ø FOR X=1 TO LA
189Ø    L$=NA$(X,6)+" "+NA$(X,1)
19ØØ    PRINT #3, SPACE$(5) +L$+
          SPACE$(LS\LA -5-LEN(L$));
191Ø NEXT X
192Ø PRINT #3,""
193Ø FOR N= 1 TO LL-3:PRINT #3,"" :NEXT N
```

Microsoft BASIC

```
194Ø IF EOF(1) THEN GOTO 196Ø
195Ø GOTO 165Ø
196Ø IF CN<CL THEN CLOSE #1:OPEN NA$ FOR
     INPUT AS#1:GOTO 164Ø ELSE CLOSE
     #1:GOTO 57Ø
2ØØØ '
2Ø1Ø ' ********************************
2Ø2Ø ' *                              *
2Ø3Ø ' *   envelope printing routine  *
2Ø4Ø ' *                              *
2Ø5Ø ' ********************************
2Ø6Ø '
2Ø7Ø PRINT:PRINT"printing envelopes"
2Ø8Ø INPUT "Width of envelope in
     characters ";EW
2Ø9Ø INPUT "Height of envelope in
     lines";EH
21ØØ ML=EW\3
211Ø FILES:INPUT"Filename for names and
     addresses";NA$:OF$=NA$:OPEN NA$ FOR
     INPUT AS #1
212Ø PRINT"Drop in an envelope and hit
     <spacebar>  to continue"
213Ø I$=INKEY$:IF I$<>CHR$(32) GOTO 213Ø
214Ø CLS
215Ø FOR N=1 TO (EH\2+12):PRINT #3,""
     :NEXT N ' 12 is number of line
     feeds to bring top of envelope in
     line with print head
216Ø FOR M=1 TO 8
217Ø    INPUT#1,NA$(1,M)
218Ø NEXT M
219Ø PRINT NA$(1,2)
22ØØ PRINT #3, SPACE$(ML);NA$(1,3)+" "
     +NA$(1,4)+" "+NA$(1,2)
221Ø PRINT #3, SPACE$(ML);NA$(1,5)
222Ø PRINT #3, SPACE$(ML);NA$(1,6)+" "
     +NA$(1,1)
223Ø FOR N=1 TO EH:PRINT #3, "":NEXT N
224Ø IF EOF(1) THEN CLOSE #1:GOTO 57Ø
225Ø GOTO 212Ø
25ØØ '
251Ø ********************************
252Ø *                              *
253Ø *   letter printing routine    *
254Ø *                              *
255Ø ********************************
256Ø '
257Ø PRINT:PRINT"printing letters"
258Ø FILES:INPUT "Filename for names and
     addresses";NA$: OF$=NA$:OPEN NA$
     FOR INPUT AS #1
259Ø FILES:INPUT "Filename for body
     text";BT$:OF$=BT$:OPEN BT$ FOR
     INPUT AS #2
26ØØ INPUT"No of blank lines above
     date";BL
261Ø INPUT "Enter today's date";DT$:IF
     BL<4 THEN SS$="":ST$="":GOTO 262Ø
2612 INPUT"Sender's Street No & Name";
     SS$
2615 INPUT"Sender's Town and
     Postcode";ST$
262Ø INPUT "Width of left margin";ML
263Ø PG=1
264Ø LC=Ø
265Ø FOR N=1 TO BL:PRINT #3, "":
     LC=LC+1:NEXT N
2655 PRINT #3,SPACE$(ML)+SS$:
     PRINT#3,SPACE$(ML )+ST$:LC=LC+2
266Ø PRINT #3, SPACE$(ML);DT$
267Ø FOR M=1 TO 8
268Ø    INPUT#1,NA$(1,M)
269Ø NEXT M
27ØØ FOR N=1 TO 5:PRINT #3,"" :
     LC=LC+1:NEXT N
271Ø PRINT #3, SPACE$(ML);NA$(1,3)+" "
     +NA$(1,4)+" "+NA$(1,2):LC=LC+1
272Ø PRINT #3,
     SPACE$(ML);NA$(1,5):LC=LC+1
273Ø PRINT #3, SPACE$(ML);NA$(1,6)+
     " "+NA$(1,1):LC=LC+1
274Ø FOR N=1 TO 3:PRINT #3,"" :
     LC=LC+1:NEXT N
275Ø PRINT #3, SPACE$(ML);"Dear "+
     NA$(1,3)+ " "+NA$(1,2)+",":LC=LC+1
276Ø FOR N=1 TO 2:PRINT #3,
     :LC=LC+1:NEXT N
277Ø LINE INPUT #2,LI$
278Ø IF LI$=CHR$(128) THEN LI$="":
     GOSUB 286Ø
279Ø IF LC>61 THEN GOSUB 286Ø
28ØØ PRINT #3, SPACE$(ML)+LI$:LC=LC+1
281Ø IF EOF(2) THEN FOR N=LC TO 65:PRINT
     #3,"":NEXT N:GOTO 282Ø ELSE GOTO
     277Ø
282Ø CLOSE #2:IF EOF(1) THEN CLOSE
     #1:GOTO 57Ø ELSE OPEN BT$ FOR INPUT
     AS#2:GOTO 263Ø
283Ø '
284Ø ' *** pagination subroutine ***
285Ø '
286Ø FOR N=LC TO 65:PRINT #3,"" :NEXT N
287Ø LC=Ø:FOR N=1 TO 5:PRINT #3,""
     :LC=LC+1:NEXT N
288Ø PG=PG+1:PRINT #3,
     SPACE$(ML+3Ø);PG:LC=LC+1
289Ø FOR N=1 TO 3:PRINT #3,"" :
     LC=LC+1:NEXT N
29ØØ RETURN
3ØØØ '
3Ø1Ø ' **** exit routine ***
3Ø2Ø '
3Ø3Ø CLOSE:END
35ØØ '
351Ø ********************************
```

Microsoft BASIC

```
3520 ' *                                *
3530 ' *    error trapping routine      *
3540 ' *                                *
3550 ' **********************************
3560 '
3570 IF ERR<>52 AND ERR<>55 THEN
     BEEP:PRINT "ERROR";ERR;"IN LINE
     #";ERL:FOR N=1 TO 1000:NEXT
     N:CLOSE:END
3580 PRINT OF$;" does not exist"
3590 PRINT"press <M> to return to menu"
3600 PRINT"press <F> to see a list of
     files "
3610 PRINT"press any other key to try
     again"
3620 AN$=INKEY$: IF AN$="" THEN 3620
3630 IF AN$="M" OR AN$="m" THEN END
3640 IF AN$<>"F" AND AN$<>"f" THEN 3680
3650 FILES
3660 PRINT"press any key to continue"
3670 AN$=INKEY$:IF AN$="" THEN 3670
3680 RESUME 570
5000 '
5010 ' **********************************
5020 ' *  INSERT YOUR EXTRA OPTION OR *
5030 ' *  SORT ROUTINE - eg sort      *
5040 ' *  from page 181 of Tandy 100  *
5050 ' *  manual after renumbering    *
5060 ' *  lines.                      *
5070 ' **********************************
5995 GOTO 570
```

Commodore 64

```
0 REM CITY BOMBER BY PAUL VANDENBERG
      FOR THE C64 WITH SUPER EXPANDER
1 FORF=3532@TO35519:READA:POKEF,A:NEXTF
2 L=10:SC=0
3 LN$="":FORF=1TO20:LN$=LN$+"  ":NEXTF
7 SPRCOL7:COLOR,,,,0:SPRITE0,0,6,,1:SPRI
TE2,0,10,,,,1
8 E=0:T=0
9 GOSUB1000
10 SPRITE0,1:SPRITE1,1
11 MOVSPR0,300,70:MOVSPR1,300,70
12 MOVSPR0,0#0:MOVSPR1,0#0
20 SPRITE0,1:SPRITE1,1
30 MOVSPR0,270#2
35 A=RSPPOS(0,0):MOVSPR1,A,+0
36 B=RSPPOS(1,1):IFB>200THEN:MOVSPR1,0#0
:SPRITE1,0:X=RSPPOS(0,0):Y=200:GOTO100
40 GETA$:IFA$<>""THEN:MOVSPR1,180#4
50 GOTO35
100 MOVSPR2,X,Y:SPRITE2,1,2
101 MOVSPR1,300,70
102 IFX<240ANDX>15THENC=INT(X/16)-1:IFB(
C)>0THENB(C)=B(C)-1:GOSUB900:GOTO104
103 IFB(C)=0THENL=L-1:L$=STR$(L):CHAR,36
,0,L$:GOSUB200
104 SPRITE2,0:SPRITE1,1:MOVSPR1,0#0
105 GOTO35
200 N=LEN(L$):CHAR,36+N,0," ":IFL=0THEN9
000
201 RETURN
899 STOP
900 SC=SC+10:GSHAPEN$,C*16,163-(B(C)*4):
IFB(C)=0THENT=T+1
901 S$=STR$(SC):CHAR,16,0,S$
902 TUNE3,0,15,3,9
903 PRINTCHR$(6)"O2V0T3U9SC"
910 IFT=14THENGOTO3000
999 RETURN
1000 GRAPHIC2,1:BOX,0,0,15,3,,1:BOX0,2,1
,3,3:BOX0,5,1,6,3:BOX0,9,1,10,3
1001 BOX0,12,1,13,3:SSHAPEZ$,0,0,15,3
1002 DIMB(13):FORF=0TO13:B(F)=INT(RND(0)
*15)+1:NEXTF
1003 SCNCLR:SSHAPEN$,0,0,15,3
1010 COLOR5,6:GRAPHIC2,1:COLOR14
1011 FORF=0TO20:CHAR,0,F,LN$:NEXTF
1012 COLOR,0:FORF=0TO13:X=F*16:AM=B(F):F
ORG=163TO(167-(AM*4))STEP-4
1013 GSHAPEZ$,X,G:NEXTG:NEXTF:L$=STR$(L)
:CHAR,30,0,"LIVES=":CHAR,36,0,L$
1014 COLOR5,6:DRAW,224,168TO319,168TO319
,185TO260,180TO224,168:PAINT,250,170,1
1015 COLOR14,0
1016 CHAR,10,0,"SCORE=":S$=STR$(SC):CHAR
,16,0,S$
1999 RETURN
2000 DATA,,,,,,,,6,,,14,,,30,127,255,255
,191,255,255,255,224,3,127,255,255
2001 DATA,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,0:REM 36 COMMAS AND ONE 0
2002 DATA,,,,,,,,,,,,,,,4,128,,7,128,,7,
128,,3,,,,,,,,,,,,,,,,,,,,,,0
2003 DATA,,,,,,,,,,,,,,,0
2004 DATA0,0,0,0,2,160,32,10,168,168,170
,168,170,169,160,42,166,128,41,86,128
2005 DATA42,86,128,10,85,168,42,85,168,1
69,86,160,165,90,128,41,86,0
2006 DATA41,86,128,9,86,160,9,169,168,42
,170,168,170,170,160,162,160,128
2007 DATA128,128,0,0,0,0,0
3000 GRAPHIC0,1
3001 PRINT"{RED}CONGRATULATIONS!!"
3002 PRINT"{BLK}YOU SUCCESSFULLY DESTROY
ED THE CITY"
3003 PRINT:PRINT"{BLUE}YOU HAD A SCORE O
F";SC
3004 PRINT"WITH";L;"LIVES LEFT"
3005 PRINT:PRINT"{WHT}WOULD YOU LIKE ANO
THER GAME?"
3006 GETA$:IFA$=""THEN3006
3007 IFA$="Y"THENCLEAR:GOTO2
9000 GRAPHIC0,1:PRINT"YOUR DEAD"
9001 PRINT"{YELO}YOU HAD A SCORE OF";SC
9002 PRINT:PRINT"{WHT}WOULD YOU LIKE ANO
THER GAME?"
9003 GETA$:IFA$=""THEN9003
9004 IFA$="Y"THENCLEAR:GOTO2
9005 PRINT"{CLR}";:END
```

CITY BOMBER

This game is for the Commodore 64 with the Super Expander. In it you fly a bomber over a city, trying to destroy it with bombs which are released by pressing any key.

You must aim carefully, as dropping a bomb in an area where the buildings have already been totally destroyed will cost you one of your lives.

Paul Vandenberg,
Cabramatta, NSW

NOTES:

Microbee

BRICKS

Bricks is the old breakout game: lots of bricks, one bat, five balls and good reflexes. This version is in machine code, has interesting sound effects, uses high-res look-a-like graphics and has 10 speed levels. The bricks slowly advance on you. The ball increases speed each round, and during each round speeds up till you hit another 32 bricks; it moves away from you at four times the speed it travels toward you (to make waiting time less). You get 1000 points if you break through the wall, and extra points for hitting more distant bricks.

To play, use the A or < key for moving up, Z or > for moving down; S is for serving; ESC holds the ball still until you release; and BRK gives you back to BASIC.

Level 0 is a good starting speed and 9 is just to show I have a sense of humour (or sadism).

This program comes in two forms, as source code and in BASIC. With the BASIC version, you type, you save, you run. The program uses low-res graphics, but I've altered the PCG's to make them look like bricks and balls. The bat is a real PCG and is not plotted on the screen.

Richard Larkin
Dee Why, NSW

```
00001REM BRICKS (BASIC VER) 10/4/85 RICHARD LARKIN
00002DATA33,0,240,17,1,240,1,0,4,54,26,237,176,33,29,44
00003DATA17,212,241,1,24,0,237,176,33,53,44,17,84,242,1,24
00004DATA0,237,176,205,6,128,33,52,45,54,0,254,89,32,2,54
00005DATA1,62,0,50,53,45,205,76,39,195,71,40,33,0,240,17
00006DATA1,240,1,191,3,54,32,237,176,33,1,240,17,2,240,1
00007DATA61,0,54,131,237,176,33,129,243,6,62,62,176,119,35,16
00008DATA252,33,127,240,17,64,0,6,13,62,183,119,25,16,252,205
00009DATA133,39,195,158,39,33,95,240,17,32,0,14,13,62,149,6
00010DATA16,119,35,35,16,251,13,25,62,0,185,32,240,201,205,39
00011DATA128,33,144,43,17,80,249,205,2,44,33,186,43,17,16,248
00012DATA205,2,44,33,170,43,17,32,248,205,2,44,33,181,43,17
00013DATA64,248,205,2,44,33,165,43,17,128,248,205,2,44,33,175
00014DATA43,17,0,249,205,2,44,33,159,43,17,0,250,205,2,44
00015DATA253,33,170,43,33,112,249,205,8,44,253,33,165,43,33,208
00016DATA249,205,8,44,253,33,159,43,33,80,251,205,8,44,62,128
00017DATA17,112,251,6,16,18,19,16,252,62,0,17,48,248,6,16
00018DATA18,19,16,252,17,0,251,6,16,18,19,16,252,62,255,50
00019DATA63,248,50,0,251,33,191,43,17,64,253,1,48,0,237,176
00020DATA33,197,43,17,112,253,1,48,0,237,176,33,202,43,17,160
00021DATA253,1,48,0,237,176,201,33,80,43,17,192,243,1,64,0
00022DATA237,176,33,77,44,17,16,242,1,12,0,237,176,205,6,128
00023DATA214,47,254,11,48,247,71,33,128,9,17,64,255,25,16,253
00024DATA34,42,45,34,44,45,33,15,242,17,16,242,1,12,0,237
00025DATA176,205,137,44,237,91,30,45,58,40,45,60,50,40,45,71
00026DATA58,41,45,254,0,40,50,184,32,47,62,0,50,40,45,58
00027DATA38,45,254,1,32,2,28,28,29,33,38,45,123,254,44,56
00028DATA10,126,254,254,40,5,54,254,205,67,42,123,254,7,48,5
00029DATA54,1,205,67,42,237,83,30,45,42,28,45,58,40,45,230
00030DATA1,254,0,32,29,58,36,45,254,1,32,2,44,44,45,125
00031DATA34,28,45,254,127,202,219,42,254,250,204,89,44,254,1,204
00032DATA104,41,42,32,45,237,91,34,45,205,51,128,42,28,45,237
00033DATA91,30,45,205,54,128,42,28,45,205,57,128,204,4,42,205
00034DATA20,41,24,52,42,46,45,17,64,0,58,48,45,6,3,119
00035DATA25,60,16,251,62,1,205,10,165,204,178,41,62,26,205,10
00036DATA165,204,224,41,62,44,205,10,165,204,178,41,62,46,205,10
00037DATA165,204,224,41,205,76,42,201,42,28,45,237,91,30,45,34
00038DATA32,45,237,83,34,45,62,48,205,10,165,40,251,62,54,205
00039DATA10,165,202,33,128,195,132,40,58,30,45,71,58,49,45,144
00040DATA79,254,7,208,254,3,62,1,50,36,45,40,7,56,2,62
```

NOTES:

Microbee

```
00041DATA254,50,38,45,6,0,33,250,43,9,126,50,41,45,33,2
00042DATA0,34,28,45,62,0,50,40,45,42,42,45,17,248,255,25
00043DATA124,254,0,200,34,42,45,33,1,0,6,40,205,135,42,16
00044DATA251,201,58,49,45,60,254,51,200,50,49,45,58,48,45,198
00045DATA3,50,48,45,254,221,192,62,212,50,48,45,42,46,45,17
00046DATA128,0,25,54,32,42,46,45,17,192,255,25,34,46,45,201
00047DATA58,49,45,61,254,5,200,50,49,45,58,48,45,214,3,50
00048DATA48,45,254,209,192,62,218,50,48,45,42,46,45,54,32,17
00049DATA64,0,24,215,42,28,45,125,203,135,15,79,214,31,203,135
00050DATA15,203,135,15,203,135,15,6,0,33,64,240,9,71,17,64
00051DATA0,14,13,126,254,133,202,102,42,254,145,202,102,42,254,148
00052DATA202,102,42,25,13,62,0,185,32,233,201,58,36,45,47,50
00053DATA36,45,201,33,20,0,6,80,205,135,42,201,42,42,45,58
00054DATA36,45,254,1,32,10,191,203,28,203,29,191,203,28,203,29
00055DATA43,124,181,32,251,201,54,128,4,205,145,42,58,53,45,60
00056DATA50,53,45,230,31,254,0,204,31,43,205,59,42,33,5,0
00057DATA1,50,0,205,135,42,201,58,52,45,254,0,200,205,95,167
00058DATA201,33,227,243,126,60,119,254,58,32,29,54,48,43,125,254
00059DATA224,32,11,58,204,243,60,254,58,40,3,50,204,243,125,254
00060DATA222,32,225,62,57,50,204,243,16,215,33,223,243,17,248,243
00061DATA6,6,78,26,185,40,4,56,6,24,15,35,19,16,243,33
00062DATA223,243,17,248,243,1,6,0,237,176,201,62,100,6,1,245
00063DATA245,205,145,42,241,33,20,0,71,205,135,42,241,61,254,0
00064DATA32,235,42,44,45,17,192,255,25,124,254,0,202,5,43,34
00065DATA44,45,34,42,45,62,0,50,53,45,33,56,0,34,32,45
00066DATA33,24,0,34,34,45,205,76,39,205,137,44,195,132,40,17
00067DATA95,240,62,13,33,2,0,25,1,30,0,237,176,33,34,0
00068DATA25,235,61,254,0,32,237,33,125,240,6,13,17,64,0,62
00069DATA149,119,25,16,252,42,28,45,125,214,2,111,34,28,45,201
00070DATA32,66,65,76,76,83,32,76,69,70,84,32,53,32,32,32
00071DATA32,32,32,32,32,32,32,32,32,83,67,79,82,69,32,48
00072DATA48,48,48,48,48,32,32,32,32,32,32,32,32,72,73,71
00073DATA72,32,83,67,79,82,69,32,48,48,48,48,48,48,32,32
00074DATA0,96,160,80,160,80,160,80,160,80,160,80,160,80,96,0
00075DATA0,0,0,0,0,0,0,0,0,0,6,9,9,9,6,0
00076DATA0,0,0,0,0,0,0,0,0,0,96,144,144,144,96,0
00077DATA0,0,0,0,0,0,0,0,0,0,192,160,144,136,136,132
00078DATA132,132,130,130,130,130,130,130,129,129,129,129,129,129,129,129
00079DATA129,130,130,130,130,130,130,132,132,132,136,136,144,160,192,0
00080DATA0,0,0,0,0,0,0,0,0,0,3,7,11,15,11,7
```

Microbee

```
00081DATA3,3,1,16,0,237,176,201,6,16,221,33,144,43,221,126
00082DATA0,253,182,0,119,221,35,253,35,35,16,242,201,32,87,69
00083DATA76,67,79,77,69,32,84,79,32,66,82,69,65,75,32,79
00084DATA85,84,32,33,32,32,87,79,85,76,68,32,89,79,85,32
00085DATA76,73,75,69,32,83,79,85,78,68,32,63,32,83,80,69
00086DATA69,68,32,40,48,45,57,41,63,58,204,243,61,254,47,40
00087DATA23,50,204,243,42,28,45-237,91,30,45,205,51,128,33,24
00088DATA0,34,30,45,205,137,44,201,205,6,128,33,248,243,17,136
00089DATA43,1,6,0,237,176,195,65,39,205,20,41,205,169,44,62
00090DATA54,205,10,165,202,33,128,62,19,205,10,165,32,235,33,56
00091DATA0,237,91,34,45,205,51,128,201,237,91,34,45,33,56,0
00092DATA205,51,128,237,91,30,45,33,56,0,205,48,128,58,38,45
00093DATA237,83,34,45,19,254,1,32,2,27,27,123,214,7,254,36
00094DATA56,14,58,38,45,254,1,62,1,32,2,62,254,50,38,45
00095DATA237,83,30,45,42,50,45,35,36,44,124,254,192,32,3,33
00096DATA0,128,34,50,45,126,230,7,33,250,43,6,0,79,9,126
00097DATA50,41,45,121,230,1,50,36,45,6,8,205,76,42,16,251
00098DATA33,56,0,34,28,45,62,0,50,40,45,201,64,0,27,0
00099DATA64,0,27,0,1,0,1,0,0,2,0,0,0,6,0,240
00100DATA218,47,0,0,1,0
00101 C=0 : FOR X=10000 TO 11573 : READ Y : POKE X,Y : C=C+Y : NEXT X
00102 IF C=1740 THEN PRINT "OK" ELSE PRINT "DATA ERROR"
00103 REM IF "OK" THEN TYPE I=USR(10000) TO PLAY GAME!
```

Hitachi Peach

TYPE.BAS

This program is a utility that emulates the CP/M's TYPE command on the Hitachi Peach. It types the required data file to the user's choice of the screen or printer, and is a useful program for viewing the contents of a data file created by the HiWriter word processing software, without having to go through the lengthy process of booting up HiWriter.

Philip Cookson,
Armadale, Vic

```
100 'TYPE.BAS
110 '-------------------------------------------------------------------
120 'Author:      Philip Cookson              Date: 04/01/85
130 'Description:
140 'This program reads a data file from the disk and types it to either
150 'the screen or a line printer.
160 '-------------------------------------------------------------------
170 '
180 GOSUB 260 'SET ERROR TRAP ON
190 GOSUB 300 'DETERMINE DATA FILE TO TYPE
200 GOSUB 350 'SELECT OUTPUT TO SCREEN OR PRINTER
210 GOSUB 430 'OPEN SPECIFIED DATA FILE
220 GOSUB 470 'READ AND PRINT THE DATA FILE
230 GOSUB 540 'CLOSE OUTPUT AND DATA FILES
240 END
250 '
260 'SUBROUTINE TO SET ERROR TRAPPING ON
270 ON ERROR GO TO 580
280 RETURN
290 '
300 'SUBROUTINE TO DETERMINE THE NAME OF THE DATA FILE TO TYPE
310 INPUT "NAME OF DATA FILE TO TYPE : ",FILENAME$
320 IF RIGHT$(FILENAME$,4)<>".DAT" THEN FILENEME$=FILENAME$+".DAT"
330 RETURN
340 '
350 'SUBROUTINE TO SELECT OUTPUT DEVICE FOR DATA FILE LISTING
360 PRINT "TYPE TO  <1> SCREEN OR <2> PRINTER ? "
370 ANS$=INKEY$:IF ANS$<>"1" AND ANS$<>"2" THEN GO TO 370
380 IF ANS$="1" THEN DEV$="SCRN:"
390 IF ANS$="2" THEN DEV$="LPT0:"
400 OPEN "O",#1,DEV$
410 RETURN
420 '
430 'SUBROUTINE TO OPEN DATA FILE
440 OPEN "I",#2,FILENAME$
450 RETURN
460 '
470 'SUBROUTINE TO READ AND PRINT THE DATA FILE
480 IF EOF(2) THEN GO TO 520
490 LINE INPUT #2, TEXT$
500 PRINT #1, TEXT$
510 GO TO 480
520 RETURN
530 '
540 'SUBROUTINE TO CLOSE THE OUTPUT AND DATA FILES
550 CLOSE #1,#2
560 RETURN
570 '
580 'SUBROUTINE TO HANDLE ERRORS
590 '
600 ' (1) File not found error
610       IF ERR=63 THEN BEEP:PRINT "FILE NOT FOUND":RESUME 230
620 '
630 ' (2) Device unavailable error
640       IF ERR=60 THEN BEEP:PRINT "DEVICE UNAVAILABLE":RESUME 230
650 '
660 ' (3) Input past end error
670       IF ERR=54 THEN RESUME 230
680 '
690 ' (4) Miscellaneous error
700       BEEP:PRINT "ERROR CODE ";ERR;" ON LINE ";ERL:RESUME 230
710 '
720 END
```

BBC

```
>L.
   10 REM **** BINGO ****
   20 REM for BBC Microcomputer
   30 REM by Syd Sanders
   40 MODE7
   50 *FX11,0
   60  *FX229,1
   65 PROCinstruct
   70 PROCinitialize
   80 PROCmain
  100 *FX12,0
  110  *FX229,0
  115 CLS:END
  120 DEFPROCinitialize
  130 DIM N%(100):R=RND(-TIME):@%=&90A:@
%=&00004:C%=0
  140 ENDPROC
  150 DEFPROCmain
  160 CLS:C%=C%+1:A%=90:B%=A%:PRINTTAB(1
3,2)CHR$(141);CHR$(131);CHR$(136)"BINGO"
:PRINTTAB(13)CHR$(141);CHR$(133);CHR$(13
6)"BINGO":PRINTTAB(0,23);CHR$(131)"GAME
"C%:P=INKEY(100)
  170 FOR I%=1 TO A%:N%(I%)=I%:NEXT
  180 FOR L%=1 TO A%
  190   IFB%=1 J%=1 ELSE J%=RND(B%)
  200   PRINTTAB(16,19)CHR$(141);CHR$(13
0)N%(J%):PRINTTAB(16)CHR$(141);CHR$(130)
N%(J%):VDU31,30,23:PRINTCHR$(134)"DRAW "
;L%:PROClot(J%)
  210     REPEAT:Z%=GET:UNTIL Z%=32 OR Z
%=27:IFZ%=27 L%=A%
  220   PRINTTAB(16,19)"         ":PRINTT
AB(16)"         "
  230   FOR K%=J%+1 TO B%:N%(K%-1)=N%(K%
):NEXT:B%=B%-1
  240   NEXT
  250 PRINTTAB(0,23);CHR$(133)"ANOTHER G
AME? ";:ANO$=FNrept:IF ANO$="Y"OR ANO$="
y"THEN 160
  260 ENDPROC
  270 DEFFNrept:test=FALSE:REPEAT:A$=GET
$:IF INSTR("YyNn",A$)=0 VDU7 ELSE test=T
RUE
  280   UNTIL test:=A$
  290 DEFPROClot(J%)
  300 IF N%(J%)MOD10=0THEN x%=36 ELSE x%
=4*((N%(J%)MOD10)-1)
  310 IF N%(J%)MOD10=0THEN y%=5+(N%(J%)D
IV10) ELSE y%=6+(N%(J%)DIV10)
  320 PRINTTAB(x%,y%)N%(J%);
  330 ENDPROC
  350 DEFPROCinstruct
  360 CLS:PRINTTAB(1,2)"Do you want inst
ructions? ";:Ins$=FNrept:IF Ins$="N"OR I
ns$="n" ENDPROC
  370 CLS:PRINTTAB(10,2)CHR$(131);"BINGO
 INSTRUCTIONS."
  380 PRINT'"When you press the SPACEBAR
 the first"
  390 PRINT"marble will be drawn and the
 result"
  400 PRINT"displayed in large green dig
its on the"
  410 PRINT"screen.  It will also be pla
ced in its"
  420 PRINT"correct position in a table
near the"
  430 PRINT"top of the screen."
  440 PRINT"To draw each additional marb
le simply"
  450 PRINT"press the SPACEBAR once."
  460 PRINT"When BINGO has been called t
he player's"
  470 PRINT"numbers can be checked again
st those"
  480 PRINT"shown in the table."
  500 PRINT"After a successful call, pre
ss ESCAPE"
  510 PRINT"to move to the next game or
to leave"
  520 PRINT"the program."
  525 PRINT"The current game number and
the number"
  526 PRINT"of marbles drawn in the curr
ent game"
  527 PRINT"are displayed at the bottom
of the"
  528 PRINT"screen at all times."
  530 PRINTTAB(5,23)"Press SPACEBAR to c
ontinue.";:REPEATUNTILGET=32:ENDPROC
>
```

BINGO

This program simulates a bingo draw. It has been written to run on the BBC Micro, but modifying it to suit other machines should not prove too difficult. The BBC version, as it stands, is well error trapped. The likelihood of an accidental crash in the middle of a game is extremely remote.

The program should prove useful to people interested in fund-raising for sporting groups or voluntary organisations.

Syd Sanders
East Victoria Park, WA

VZ 200

NUMBER SLIDE

Number slide is a computer version of the puzzles that used to be given away with breakfast cereal. This version has been adapted from a ZX81 program printed in this magazine a few years ago.

The idea is to rearrange the numbers in correct order after the computer has mixed them up. The program should work on other computers without much modification.

Bruce Daniel
Mudgee NSW

```
10 DIMA(9):BS$=CHR$(8)+CHR$(8)+CHR$(127)+CHR$(127)
12 Z$="
15 CLS
20 FORX=1TO9
30 LETA(X)=0
40 NEXTX
50 LETA(5)=-32
60 FORX=1TO9
70 IFX=5THENGOTO130
80 LETP=RND(8)
90 FORY=1TO9
100 IFA(Y)=PTHENGOTO80
110 NEXTY
120 LETA(X)=P
130 NEXTX
140 PRINT@224,Z$;Z$;:PRINT@0,;
200 FORX=1TO3
210 FORY=1TO3
220 PRINTCHR$(A(Y+(X-1)*3)+64);" ";
240 NEXTY
250 IFX=1THENPRINT"   123"
260 IFX=2THENPRINT"   456"
270 IFX=3THENPRINT"   789"
280 PRINT
290 NEXTX
300 PRINT
310 PRINT@256,"MOVE FROM:   ";BS$;
320 INPUTF
340 IFF>9THENGOTO310
350 PRINT
360 PRINT@320,"MOVE TO:   ";BS$;
370 INPUTT
380 PRINT:IF(F=3ANDT=4)OR(F=4ANDT=3)ORT=0THEN310
390 IFT>9OR(F=6ANDT=7)OR(F=7ANDT=6)THENGOTO360
400 IFNOTA(T)=-32THENGOTO360
410 IFABS(F-T)=1ORABS(F-T)=3THENGOTO430
420 GOTO310
430 LETA(T)=A(F)
440 LETA(F)=-32
450 CLS
470 FORI7=1TO7
480 IFA(I7)>A(I7+1)THENGOTO200
490 NEXTI7
500 PRINT"CONGRATULATIONS, "
510 PRINTTAB(5)"YOU HAVE SOLVED THE PUZZLE."
520 PRINT
530 INPUT"TRY AGAIN (Y/N) ";X$
540 IFX$<>"N"THENRUN
550 CLS:END
```

VZ200

ELECTRIC TUNNEL

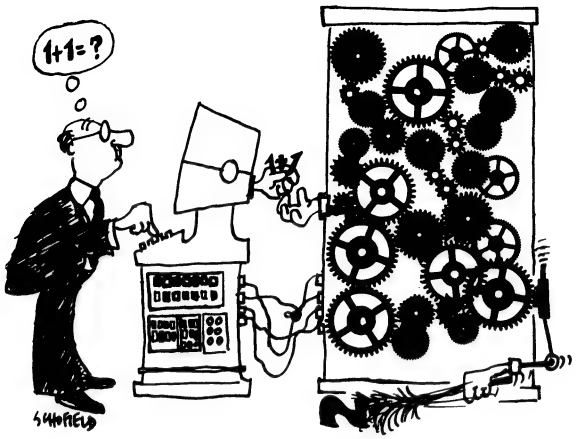
The object of the game is to travel along the tunnel, avoiding the electrically charged walls.

The program uses joysticks for control, but by modifying lines 170 and 180 the program could use the keyboard:
170 KY$=INKEY$
180 IF KY$="M" THEN Z=Z-1 ELSE IF KY$="," THEN=Z Z+1

The PEEK in line 190 checks to see if the position in front of you is clear. Scoring is based on the distance you travel along the tunnel.

Bruce Daniel,
Mudgee, NSW

```
0 '  ELECTRIC TUNNEL
1 ' WRITTEN BY    BRUCE DANIEL
2 '
10 CLS : COLOR 2,0
20 P$ = CHR$(143)
30 FOR I=1 TO 10 : P$=P$+CHR$(176)
40 NEXT I:P$=P$+CHR$(143)
50 IF INKEY$<>"" THEN X=RND(0) :GOTO 50
100 PP=16-INT(LEN(P$)/2)
110 Z=16
130 PRINT TAB(PP);P$ :POKE 28672+Z,99
140 IF RND(2)=1 THEN PP=PP+RND(3)-2
150 IF PP<3 THENPP=3ELSE IFPP>(32-LEN(P$)-3)THENPP=32-LEN(P$)-3
160 IF CN<16 THEN 290
170 JK= INP(43) AND INP(46) AND 31
180 IF JK=27 THEN Z=Z-1ELSE IF JK=23 THEN Z=Z+1
190 L=PEEK(28704+Z):IF L<>144 AND L<>176 AND L<>128 THEN 400
290 CN=CN+1:IF CN/30<>INT(CN/30) THEN 130
300 Q=LEN(P$)
310 IF Q<=5 THEN 130
320 P$=LEFT$(P$,1)+MID$(P$,2,Q-3)+RIGHT$(P$,1)
330 GOTO 130
400 PRINT:POKE 28672+Z,45
410 COLOR,1:SOUND31,1:SOUND31,1:SOUND23,1:SOUND23,1
420 SOUND13,1:SOUND13,1:SOUND4,5
425 '
440 SOUND 0,2
450 COLOR,0
460 FORI=1TO5
470 FORTD=1TO25:NEXTTD
480 PRINT@0,"-++ CRASH CRASH CRASH CRASH ++- ";
490 FORTD=1TO25:NEXTTD
500 PRINT@0,"                                ";
510 FORTD=1TO25:NEXTTD,I
520 PRINT@128,"SCORE:";'INVERSE 'SCORE'
530 SC=INT(CN*1.2-DN):PRINTSC;
540 PRINT@480,"  PRESS <RETURN> TO TRY AGAIN";
550 IF INKEY$<>CHR$(13) THEN 550
560 RUN
```



BBC

LOAN PRINT

The program will print any reducible interest loans either to the screen only or to both the screen and the printer. It has a page mode built in so that, if the loan details exceed one screen, those visible can be viewed before going on to the next page. Use the shift key to go to the next page.

If the printer being used does not auto-linefeed, line 1080 should have the following added: :*FX6 after the statement THEN VDU2.

I have found the program most useful when comparing different loans from various lending bodies, and have saved thousands of dollars in interest.

Kenneth Nicholls,
Kanahooka, NSW

```
10REM*****************************
20REM**                         **
30REM**   LOAN-PRINT            **
40REM**   By John Nicholls      **
50REM**   Version 3.3           **
60REM**   Prog subject to       **
70REM**   Copyright             **
80REM**   Mag Name 1984         **
90REM**                         **
100REM*****************************
110MODE7
120ON ERROR PROCerror
130PROCinstruct
140PROCquest
150MODE3
160PROCinit
170PROCloan
180PROCcalc
190PRINT"  The Monthly Payment is            $";amount
200PROCsheet
210PROCmain
220INPUT"  Process Another Loan ( YES/NO)";A$
230IF A$="YES" OR A$="Y" THEN 160
240VDU3
250MODE6
260END
270DEFPROCinit
290@%=&2020E
300VDU23;8202;0;0;0;
310VDU14
320COLOUR129
330COLOUR0
340CLS
350ENDPROC
360DEFPROCloan
370INPUT"  Enter Amount of the Loan          ";loan
380INPUT"  Enter the Annual Interest Rate    ";rate
390INPUT"  Enter the Term in Years           ";years
400ENDPROC
410DEFPROCcalc
420interest=rate/12
430month=years*12
440amount=(interest*loan)/((1-(interest+1)^-month))
450result=amount
460PROCroundoff
470amount=result
480ENDPROC
490DEFPROCroundoff
500result=INT(100*(result-0.005))/100
510ENDPROC
520DEFPROCsheet
530balance=loan
```

BBC

```
 540total=0
 550PRINTTAB(10)"MONTH";:PRINTTAB(22)"TO";:PRINTTAB(36)"TO"
 560PRINTTAB(10)"NUM";:PRINTTAB(22)"INTEREST";:PRINTTAB(36)"PRINCPL";:PRINTTAB(
8)"PRIN BAL";:PRINTTAB(63)"TOT INT"
 570ENDPROC
 580DEFPROCinterest
 590inter1=balance*interest
 600result=inter1
 610PROCroundoff
 620ENDPROC
 630DEFPROCmain
 640FORline=1 TO month
 650PROCinterest
 660inter1=result
 670print=amount-inter1
 680IF line=month THEN print=balance
 690balance=balance-print
 700total=total+inter1
 710PRINTline,inter1,print,balance,total
 720NEXTline
 730ENDPROC
 740DEFPROCinstruct
 750VDU 23;8202;0;0;0;
 760PRINT CHR$(141);"            LOAN PRINTOUT"
 770PRINT CHR$(141);"            LOAN PRINTOUT"
 780PRINT''''
 790PRINTTAB(3);CHR$(131);"This program is designed to "
 800PRINTTAB(3);CHR$(131);"print out a reducible interest "
 810PRINTTAB(3);CHR$(131);"loan. All the instructions are "
 820PRINTTAB(3);CHR$(131);"simple to follow, but for example"
 830PRINTTAB(3);CHR$(131);"when asked to enter an"
 840PRINTTAB(3);CHR$(131);"interest rate of 25% or 60%"
 850PRINTTAB(3);CHR$(131);"it should be entered 0.25 or 0.60"
 860PRINTTAB(3);CHR$(131);"or whatever rate is appropriate."
 870PRINTTAB(3);CHR$(131);"If the term of the loan is"
 880PRINTTAB(3);CHR$(131);"greater than 18 months then"
 890PRINTTAB(3);CHR$(131);"the program enters page mode"
 900PRINTTAB(3);CHR$(131);"press shift to scroll page"
 910PRINT''
 920PRINTCHR$(129);"       Press any key to continue"
 930K%=GET
 940ENDPROC
 950DEFPROCquest
 960CLS
 970PRINTTAB(3);CHR$(131);"I.E.enter the data as follows"
 980PRINT
 990PRINTTAB(3);CHR$(131);"Enter amount of Loan ?";:PRINT CHR$(133);"1000"
1000PRINT
1010PRINTTAB(3);CHR$(131);"Enter the Annual Interest Rate?";:PRINTCHR$(133);"0.
15"
1020PRINTTAB(3);CHR$(131);"Enter the Term in Years ?";:PRINTCHR$(133);"3"
1030PRINT''''''''''
1040PRINTTAB(3);CHR$(131);"Do you wish to use a printer ?"
1050PRINTTAB(3);CHR$(131);"Enter YES or NO"
1060PRINT'
1070PRINT"   ";:INPUT P$
1080IF P$="YES"ORP$="Y" OR P$="yes" OR P$="y" THEN VDU2
1090ENDPROC
1100DEFPROCerror
1110CLS                  1140PRINT" at line "ERL
1120VDU3                 1150END
1130REPORT               1160ENDPROC
```

BBC

A sample run of the Loan Print program.

```
Enter Amount of the Loan          ?20000
Enter the Annual Interest Rate    ?0.156
Enter the Term in Years           ?4
The Monthly Payment is            $562.72
```

MONTH NUM	TO INTEREST	TO PRINCPL	PRIN BAL	TOT INT
1.00	260.00	302.72	19697.28	260.00
2.00	256.06	306.66	19390.62	516.06
3.00	252.08	310.64	19079.98	768.14
4.00	248.04	314.68	18765.30	1016.18
5.00	243.95	318.77	18446.53	1260.13
6.00	239.80	322.92	18123.61	1499.93
7.00	235.61	327.11	17796.50	1735.54
8.00	231.35	331.37	17465.13	1966.89
9.00	227.05	335.67	17129.46	2193.94
10.00	222.68	340.04	16789.42	2416.62
11.00	218.26	344.46	16444.96	2634.88
12.00	213.78	348.94	16096.02	2848.66
13.00	209.25	353.47	15742.55	3057.91
14.00	204.65	358.07	15384.48	3262.56
15.00	200.00	362.72	15021.76	3462.56
16.00	195.28	367.44	14654.32	3657.84
17.00	190.51	372.21	14282.11	3848.35
18.00	185.67	377.05	13905.06	4034.02
19.00	180.77	381.95	13523.11	4214.79
20.00	175.80	386.92	13136.19	4390.59
21.00	170.77	391.95	12744.24	4561.36
22.00	165.68	397.04	12347.20	4727.04
23.00	160.51	402.21	11944.99	4887.55
24.00	155.28	407.44	11537.55	5042.83
25.00	149.99	412.73	11124.82	5192.82
26.00	144.62	418.10	10706.72	5337.44
27.00	139.19	423.53	10283.19	5476.63
28.00	133.68	429.04	9854.15	5610.31
29.00	128.10	434.62	9419.53	5738.41
30.00	122.45	440.27	8979.26	5860.86
31.00	116.73	445.99	8533.27	5977.59
32.00	110.93	451.79	8081.48	6088.52
33.00	105.06	457.66	7623.82	6193.58
34.00	99.11	463.61	7160.21	6292.69
35.00	93.08	469.64	6690.57	6385.77
36.00	86.98	475.74	6214.83	6472.75
37.00	80.79	481.93	5732.90	6553.54
38.00	74.53	488.19	5244.71	6628.07
39.00	68.18	494.54	4750.17	6696.25
40.00	61.75	500.97	4249.20	6758.00
41.00	55.24	507.48	3741.72	6813.24
42.00	48.64	514.08	3227.64	6861.88
43.00	41.96	520.76	2706.88	6903.84
44.00	35.19	527.53	2179.35	6939.03
45.00	28.33	534.39	1644.96	6967.36
46.00	21.38	541.34	1103.62	6988.74
47.00	14.35	548.37	555.25	7003.09
48.00	7.22	555.25	0.00	7010.31

```
Process Another Loan ( YES/NO)?N
```

Microbee

MUSICAL MICROBEE

The program assigns a note value to each key in the top row of the keyboard (excluding the black ones). Pressing the key sounds the note, and writes it to the screen as a crotchet (or crotchet rest) on the displayed staff. The range is about an octave and a half. Bar lines are displayed every four notes, and a piece can contain up to 109 notes.

Displayed note sequences may be played or edited. Pressing RETURN starts the music playing, as does LINEFEED, but at a slower speed. During play, a cursor moves along the piece.

The piece may be changed by 'overwriting' at the cursor position. The cursor may be shifted using:
< left
> right
ESC to start of piece
BACKSPACE to end of piece

In addition, a note may be deleted using the DEL key.

The limitations of Musical Microbee are that it only plays crotchets and whole tones, has limited note range and only two playing speeds. Despite this, the program held the interest of my children for a reasonable length of time. It was meant as a first exposure to musical notation.

Drew Krix,
Kaleen, ACT

```
      M U S I C  Program
      ==================

0100 GOSUB 490:POKE 220,5:PCG
0110 CURS T(X,0),T(X,2)+2:PRINT" ";
0120 A4$=KEY:IF A4$="" THEN 120
0130 IF A4$="." THEN 280
0140 IF A4$="," THEN 300
0150 IF ASC(A4$)=8 THEN 320
0160 IF ASC(A4$)=13 THEN 370
0170 IF ASC(A4$)=10 THEN 390
0180 IF ASC(A4$)=27 THEN 340
0190 IF ASC(A4$)=127 THEN 410
0200 REM ** put in note
0210 L=0:FOR K=1 TO 13:IF N1$(;K,K)=A4$
     THEN LET L=K
0220 NEXT K:IF L=0 THEN 110 ELSE PLAY N(
     L,0),2
0230 IF T(X,1)=100 THEN IF X<M THEN LET
     T(X+1,1)=100
0240 T(X,1)=L:CURS T(X,0),T(X,2):PRINT"8
     ";A1$(;N(L,1),N(L,1));
0250 CURS T(X,0),T(X,2)+1:PRINT"9";A1$(;
     N(L,2),N(L,2));:IF L=2 THEN CURS T(X
     ,0),T(X,2)+1:PRINT"^";
0260 X=X+1:IF X>M THEN LET X=M
0270 GOTO 110
0280 REM move cursor right
0290 IF T(X,1)=100 OR X=M THEN 110 ELSE
     LET X=X+1:GOTO 110
0300 REM move cursor left
0310 IF X=0 THEN 110 ELSE LET X=X-1:GOTO
     110
0320 REM return to 1st position
0330 X=0:GOTO 110
0340 REM go to end
0350 FOR X=0 TO M:IF T(X,1)=100 THEN NEX
     T*X 110: ELSE NEXT X
0360 GOTO 110
0370 REM play tune
0380 FOR K=0 TO M:IF T(K,1)=100 THEN NEX
     T*K 110: ELSE CURS T(K,0),T(K,2)+2:P
     RINT" ";:PLAY N(T(K,1),0),2:NEXT K:G
     OTO 110
0390 REM play slo
0400 FOR K=0 TO M:IF T(K,1)=100 THEN NEX
     T*K 110: ELSE CURS T(K,0),T(K,2)+2:P
     RINT" ";:PLAY N(T(K,1),0),3:PLAY 0,1
     :NEXT K:GOTO 110
0410 REM delete note
0420 IF T(X,1)=100 THEN 110 ELSE IF X=M
     THEN LET T(X,1)=100:K=X:GOTO 480
0430 FOR J=X TO M-1:T(J,1)=T(J+1,1):L=T(
     J,1)
0440 IF L=100 THEN LET K=J:NEXT*J 480
0450 CURS T(J,0),T(J,2):PRINT"8";A1$(;N(
     L,1),N(L,1));
0460 CURS T(J,0),T(J,2)+1:PRINT"9";A1$(;
     N(L,2),N(L,2));
0470 NEXT J:T(M,1)=100:K=M
0480 CURS T(K,0),T(K,2):PRINT"88";:CURS
     T(K,0),T(K,2)+1:PRINT"99":GOTO 110
0490 REM **** initialize ******
0500 POKE 220,20:IN#6 ON:M=108:DIM T(M,2
     ),N(13,2)
0510 CLS:CURS 21,8:PRINT"* * * M U S I C
     * * *";
0520 A1$="0123456789abcdefghijklmnopqrst
     uvwxyz:-^ "
0530 PCG:FOR K=1 TO LEN(A1$)
0540 P=63488+ASC(A1$(;K,K))*16:FOR I=P T
     O P+15:READ D:POKE I,D:NEXT I
0550 NEXT K
0560 CURS 2,1:PRINT"208468";:CURS 2,2:PR
     INT"319579";
0570 CURS 8,1:PRINT"88a";:FOR K=1 TO 6:P
     RINT"88888888a";:NEXT K
0580 CURS 8,2:PRINT"99b";:FOR K=1 TO 6:P
     RINT"99999999b";:NEXT K
0590 FOR J=4 TO 10 STEP 3:CURS 2,J:FOR K
     =1 TO 7:PRINT"88888888a";:NEXT K
0600 CURS 2,J+1:FOR K=1 TO 7:PRINT"99999
     999b";:NEXT K:NEXT J
0610 X=0:T(0,1)=100
0620 L=0:T(L,2)=1:T(L,0)=8:FOR K=1 TO 6:
     L=L+1:T(L,2)=1:T(L,0)=T(L-1,0)+3:FOR
     J=1 TO 3:L=L+1:T(L,2)=1:T(L,0)=T(L-1
     ,0)+2:NEXT J:NEXT K
0630 FOR Y=4 TO 10 STEP 3:T(L,0)=-1:FOR
     K=1 TO 7:L=L+1:T(L,2)=Y:T(L,0)=T(L-1
     ,0)+3:FOR J=1 TO 3:L=L+1:T(L,2)=Y:T(
     L,0)=T(L-1,0)+2:NEXT J:NEXT K:NEXT Y
     :T(24,0)=62:T(52,0)=62:T(80,0)=62
0640 FOR J=1 TO 13:READ N(J,0):NEXT J:FO
     R J=1 TO 13:READ N(J,1):NEXT J
0650 FOR J=1 TO 13:READ N(J,2):NEXT J
0660 REM **(above)set up n( ,1) =top cha
     r no; ,2) = bottom
0670 REM set up keys to produce the note
     s
0680 N1$="1234567890:-^"
0690 REM print prompts
0700 L=-4:FOR J=1 TO 13:L=L+5:CURS L,14:
     K=11+J*2:PCG:PRINT"8";A1$(;K,K);"8";
     :CURS L,15:PRINT"9";A1$(;K+1,K+1);"9
     ";:CURS L,16:NORMAL:PRINT"<";N1$(;J,
     J);">";:NEXT J
0710 CURS 6,15:PCG:PRINT"^";
0720 NORMAL:CURS 1,13
0730 PRINT"Move cursor: <, >, BK-SP, ESC
     Play: LN-FD, RTN    Delete: DEL"
0740 RETURN
0750 REM Chars are 0,1,2,3,4,5,6,7,8,9,a
     ,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s
     ,t,u,v,w,x,y,z,:,-,^
0760 DATA 96, 240, 208, 136, 12, 255, 12
     , 28, 56, 112, 255, 224, 160, 32, 32
     , 255
0770 DATA 254, 147, 17, 17, 255, 17, 17,
     18, 148, 255, 8, 8, 136, 200, 200, 2
     40
0780 DATA 0, 0, 0, 1, 1, 255, 1, 1, 1, 0
     , 255, 1, 3, 7, 14, 255
0790 DATA 28, 56, 113, 97, 255, 17, 8, 4
     , 2, 255, 0, 0, 3, 7, 7, 3
0800 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
     , 255, 2, 12, 24, 24, 255
0810 DATA 24, 28, 14, 3, 255, 0, 0, 0, 0
     , 255, 0, 0, 0, 0, 0, 0
0820 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
     , 255, 152, 4, 60, 60, 255
0830 DATA 0, 0, 4, 136, 255, 0, 0, 0, 0,
     255, 0, 0, 0, 0, 0, 0
0840 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
     , 255, 0, 0, 0, 0, 255
0850 DATA 0, 0, 0, 0, 255, 0, 0, 0, 0, 2
     55, 0, 0, 0, 0, 0, 0
0860 DATA 0, 0, 0, 0, 0, 255, 1, 1, 1, 1
     , 255, 1, 1, 1, 1, 255
0870 DATA 1, 1, 1, 1, 255, 1, 1, 1, 1, 2
     55, 0, 0, 0, 0, 0, 0
0880 DATA 0, 0, 0, 0, 0, 255, 0, 0, 32,
     16, 255, 4, 14, 28, 56, 255
0890 DATA 48, 16, 8, 12, 255, 50, 48, 24
     , 4, 255, 0, 0, 0, 0, 0, 0
0900 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
     , 255, 0, 0, 4, 4, 255
0910 DATA 4, 4, 4, 4, 255, 4, 4, 4, 4, 2
     55, 4, 4, 116, 252, 255, 120
0920 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
```

Microbee

```
    , 255, 4, 4, 4, 4, 255
0930 DATA 4, 4, 4, 4, 255, 4, 4, 4, 4, 2
    55, 116, 252, 252, 120, 0, 0
0940 DATA 0, 0, 0, 0, 0, 255, 0, 0, 4, 4
    , 255, 4, 4, 4, 4, 255
0950 DATA 4, 4, 4, 4, 255, 4, 4, 116, 25
    2, 255, 252, 120, 0, 0, 0, 0
0960 DATA 0, 0, 0, 0, 0, 255, 4, 4, 4, 4
    , 255, 4, 4, 4, 4, 255
0970 DATA 4, 4, 4, 4, 255, 116, 252, 252
    , 120, 255, 0, 0, 0, 0, 0, 0
0980 DATA 0, 0, 0, 4, 4, 255, 4, 4, 4, 4
    , 255, 4, 4, 4, 4, 255
0990 DATA 4, 4, 116, 252, 255, 252, 120,
    0, 0, 255, 0, 0, 0, 0, 0, 0
1000 DATA 0, 4, 4, 4, 4, 255, 4, 4, 4, 4
    , 255, 4, 4, 4, 4, 255
1010 DATA 116, 252, 252, 120, 255, 0, 0,
    0, 0, 255, 0, 0, 0, 0, 0, 0
1020 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
    , 255, 0, 0, 120, 252, 255
1030 DATA 252, 116, 4, 4, 255, 4, 4, 4,
    4, 255, 4, 4, 4, 4, 4, 4
1040 DATA 0, 0, 0, 0, 0, 255, 0, 0, 0, 0
    , 255, 120, 252, 252, 124, 255
1050 DATA 4, 4, 4, 4, 255, 4, 4, 4, 4, 2
    55, 4, 4, 4, 4, 0, 0
1060 DATA 0, 0, 0, 0, 0, 255, 0, 0, 120,
    252, 255, 252, 116, 4, 4, 255
1070 DATA 4, 4, 4, 4, 255, 4, 4, 4, 4, 2
    55, 4, 4, 0, 0, 0, 0
1080 DATA 0, 0, 0, 0, 0, 255, 120, 252,
    252, 116, 255, 4, 4, 4, 4, 255
1090 DATA 4, 4, 4, 4, 255, 4, 4, 4, 4, 2
    55, 0, 0, 0, 0, 0, 0
1100 DATA 0, 0, 0, 120, 252, 255, 252, 1
    16, 4, 4, 255, 4, 4, 4, 4, 255
1110 DATA 4, 4, 4, 4, 255, 4, 4, 0, 0, 2
    55, 0, 0, 0, 0, 0, 0
1120 DATA 0, 120, 252, 252, 116, 255, 4,
    4, 4, 4, 255, 4, 4, 4, 4, 255
1130 DATA 4, 4, 4, 4, 255, 0, 0, 0, 0, 2
    55, 0, 0, 0, 0, 0, 0
1140 DATA 0, 0, 0, 0, 255, 0, 0, 0, 0, 2
    55, 0, 0, 0, 0, 15, 0
1150 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0
1160 REM ** data to set up n( , ) array
1170 DATA  0, 4, 6, 8, 9,11,13,15,16,18,
    20,21,23
1180 DATA 13,15,17,19,21,23,25,27,29,31,
    33,35,37
1190 DATA 14,16,18,20,22,24,26,28,30,32,
```

NOTES:

Commodore 64

DIAMOND MINER

This program lets you play Diamond Dan, a diamond miner. Your boss wants you to mine for as many diamonds as possible, and to speed you up a bit he has imposed a time limit within which to collect individual stones. The limit starts at 14 seconds, but if you get too fast — picking up a diamond in, say, five seconds — the boss may lower your time limit.

You must also be careful to collect only the white diamonds; the green ones are booby-trapped and will cost you a life. Watch out for rocks, as they will also reduce your life expectancy. The worst of all dangers are the walls: bumping into them will kill you instantly.

Added to all these dangers are red crosses which, when landed on, will move you randomly to another area of the mine.

With this game you can plug the joystick into either port. If you don't have a joystick, use the following keys instead:

A = left
D = right
W = up
X = down

The codes listed below substitute for graphics characters on the Commodore.

(C/RT) = Cursor Right
(C/DN) = Cursor Down
(RVON) = Reverse On
(RVOF) = Reverse Off
(PURP) = Purple
(LRED) = Light-red
(LGRN) = Light-green
(LBLU) = Light-blue
(YELO) = Yellow
(CLR) = Clear
(HOME) = Home

Paul Vandenberg
Cabramatta NSW

```
0 REM DIAMOND MINER BY PAUL VANDENBERG
1 POKE53280,0:POKE53281,0:PRINT"{PURP}"
2 CH=5:LY=20:LX=20
3 TI$="000000":TL=900
4 D$="{C/DN}":R$="{C/RT}"
5 JS=56320:GOSUB2000
6 S=54272:FORZ=STOS+24:POKEZ,0:NEXT:POKE
S+5,9:POKES+6,0:POKES+24,15
100 PRINT"{CLR}";
101 PRINT"{RVON}
                ";
102 FORF=1TO20:PRINT" ";SPC(38);" ";:NEX
TF
103 PRINT"
          {RVOF}"
105 CL=1:FORF=1TO5:X=INT(RND(0)*38)+1:Y=
INT(RND(0)*20)+1:C=90:GOSUB10000
106 NEXTF
109 FORF=1TO15:A=PEEK(197):J1=PEEK(JS):J
2=PEEK(JS+1):POKES+4,32
110 POKES+1,50:POKES+4,129:POKES+4,0:IFT
I>=TLTHENGOSUB8000
111 CL=0:C=32:X=LX:Y=LY:GOSUB10000:IFA=9
ORJ1=126ORJ2=254THENLY=LY-1
112 IFA=230RJ1=125ORJ2=253THENLY=LY+1
113 IFA=100RJ1=123ORJ2=251THENLX=LX-1
114 IFA=180RJ1=119ORJ2=247THENLX=LX+1
115 P=LY*40+LX:L=PEEK(55296+P)
116 POKES+4,32:IFL=4THEN9000
117 IFL=6THENCH=CH-1:POKES+1,8:POKES+4,3
3:FORZ=1TO50:NEXTZ:IFCH=0THEN9000
118 IFL=1THENPT=PT+100:CT=TI:TI$="000000
":GOSUB5000:IFCT>360THENTL=TL-20
119 IFL=5THENCH=CH-1:POKES+1,8:POKES+4,3
3:FORZ=1TO50:NEXTZ:IFCH=0THEN9000
120 IFL=10THENGOSUB5050:LX=INT(RND(0)*38
)+1:LY=INT(RND(0)*20)+1
121 CL=14:C=42:X=LX:Y=LY:GOSUB10000
122 IFTI$="000015"THENGOSUB8000
124 PRINT"{HOME}{RVON}{PURP}"TI$;TAB(10)
;"SCORE";PT;TAB(26);"MEN LEFT";CH
129 NEXTF
130 CL=1:FORF=1TO1:X=INT(RND(0)*38)+1:Y=
INT(RND(0)*20)+1:C=90:GOSUB10000
131 CL=6:FORF=1TO4:X=INT(RND(0)*38)+1:Y=
INT(RND(0)*20)+1:C=224:GOSUB10000:NEXTF
132 CL=5:X=INT(RND(0)*38)+1:Y=INT(RND(0)
*20)+1:C=90:GOSUB10000
133 IFRND(0)>.8THENCL=10:X=INT(RND(0)*38
)+1:Y=INT(RND(0)*20)+1:C=86:GOSUB10000
160 GOTO109
2000 REM TITLE PAGE
2001 PRINT"{CLR}";:FORZ=1TO5:PRINTD$;:NE
XTZ
2002 PRINT"{PURP}";
2003 PRINT"{RVON}  ◥{C/RT}   {C/RT}◤ ◥{C
/RT}◥{C/RT}{C/RT}◤{C/RT}◤ ◥{C/RT}◥{C/RT}
{C/RT} {C/RT}  ◥"
2004 PRINT"{RVON} {C/RT} {C/RT}{C/RT} {C
/RT}{C/RT} {C/RT} {C/RT} {RVOF}╲╱{RVON}
{C/RT} {C/RT} {C/RT} {RVOF}╲{RVON}{C/RT}
 {C/RT} {C/RT} "
2005 PRINT"{RVON} {C/RT} {C/RT}{C/RT} {C
/RT}{C/RT}   {C/RT} {C/RT}{C/RT} {C/RT}
{C/RT} {C/RT} {C/RT}{RVOF}╲{RVON} {C/RT}
 {C/RT} "
2006 PRINT"{RVON} {C/RT} {C/RT}{C/RT} {C
/RT}{C/RT} {C/RT} {C/RT} {C/RT}{C/RT} {C
/RT} {C/RT} {C/RT} {C/RT}{C/RT} {C/RT} {
C/RT} "
2007 PRINT"{RVON}  {RVOF}◤ {RVON}   {C/R
T} {C/RT} {C/RT} {C/RT}{C/RT} {C/RT}{RVO
F}◥{RVON} {RVOF}◤{C/RT}{RVON} {RVOF} {C/
RT}{RVOF}◥ {RVON}  {RVOF}◤"
2008 PRINT:PRINT
2009 PRINTTAB(10);"{RVON}◥{C/RT}{C/RT}◤{
C/RT}   {C/RT}◥{C/RT}{C/RT} {C/RT}   {C/
RT} ◥"
2010 PRINTTAB(10);"{RVON} {RVOF}╲╱{RVON}
 {C/RT}{C/RT} {C/RT}{C/RT} {RVOF}╲{RVON}
{C/RT} {C/RT} {C/RT}{C/RT}{C/RT} {C/RT}
"
2011 PRINTTAB(10);"{RVON} {C/RT}{C/RT} {
C/RT}{C/RT} {C/RT}{C/RT} {C/RT}{RVOF}╲{R
VON} {C/RT}  {C/RT}{C/RT}  {RVOF}◤"
2012 PRINTTAB(10);"{RVON} {C/RT}{C/RT} {
C/RT}{C/RT} {C/RT}{C/RT} {C/RT}{C/RT} {C
/RT} {C/RT}{C/RT}{C/RT} {RVOF}╲{RVON}"
2013 PRINTTAB(10);"{RVON} {C/RT}{C/RT} {
C/RT}   {C/RT} {C/RT}{C/RT}{RVOF}◥ {RVON
}   {C/RT} {C/RT}{RVOF}╲"
2015 PRINT:PRINT"{C/RT}{C/RT}{C/RT}{C/RT
}{C/RT}{LGRN}BY PAUL VANDENBERG"
2016 PRINT:PRINT"{LRED}PRESS A KEY TO ST
ART{PURP}"
2017 GETA$:IFA$=""THEN2017
2018 RETURN
5000 POKES+1,50:POKES+4,17:POKES+4,16:FO
RZ=1TO20:NEXTZ

5001 POKES+1,60:POKES+4,17:POKES+4,16:FO
RZ=1TO20:NEXTZ
5002 RETURN
5050 FORF=40TO70STEP2:POKES+1,F:POKES+4,
33:NEXTF:POKES+4,32:RETURN
7999 REM NEXT LEVEL MARKER""
8000 REM TIME UP
8001 PRINT"{HOME}";:FORZ=1TO23:PRINTD$;:
NEXTZ:PRINT"{PURP}YOU WERE A BIT SLOW"
8002 POKES+1,6:POKES+4,33:FORZ=1TO2000:N
EXTZ:CH=CH-1:IFCH=0THEN9000
8003 PRINT"{C/UP}                    {HOM
E}":POKES+4,32
8004 TI$="000000":RETURN
8500 RETURN
9000 PRINT"{CLR}YOU'RE DEAD!!"
9001 GETA$:IFA$<>""THEN9001
9002 PRINT"{LBLU}YOU'VE ACCUMULATED A TO
TAL SCORE OF"
9003 PRINT:FORZ=1TO20:PRINT"{C/RT}{C/RT}
{C/RT}{C/RT}{C/UP}{YELO}";PT:FORG=1TO40:
NEXTG:PRINT"{C/RT}{C/RT}{C/RT}{C/RT}{C/U
P}{GRN}";PT
9004 POKES+1,5:POKES+4,33:POKES+4,32
9005 FORG=1TO38:NEXTG:NEXTZ
9006 PRINT:PRINT"{LGRN}WOULD YOU LIKE AN
OTHER GO? (Y/N)"
9007 GETA$:IFA$=""THEN9007
9008 IFA$="Y"THENCLR:GOTO1
9009 END
10000 P=Y*40+X:POKEP+1024,C:POKE55296+P,
CL:RETURN
```

Commodore 16

COMPATIBILITY ANALYSIS

The program displays a compatibility table for two people based on the biorhythmic cycle theory. The table consists of physical cycle, sensitivity cycle and cognitive cycle compatibility.

Although it was written on a C16, the program should also run without modification on the Plus Four.

Mark Wilkinson
Heyfield VIC

```
10 REM ***************************
20 REM *       COMMODORE 16      *
30 REM * COMPATABILITY ANALYSIS *
40 REM *    AUTHOR:M.WILKINSON   *
50 REM ***************************
60 REM
70 DIMA1(30),B1(30),M$(12),D$(7),A(12)
80 COLOR0,3,2:COLOR4,6,3
90 FORI=1TO12:READA(I):NEXT
100 DATA0,31,59,90,120,151,181,212,243,273,304,334
110 GOSUB800
120 Y=0:FO$="%###.##"
130 Y=Y+1
140 INPUT"■■■NAME OF PERSON 1.....■";W$
150 INPUT"■BIRTHDAY (DD,MM,YYYY).■";D,M,Y
160 E1=M:F1=D:G1=Y:GOSUB600
170 Z2=T:K1=J+1
180 INPUT"■■■NAME OF PERSON 2.....■";X$
190 INPUT"■BIRTHDAY (DD,MM,YYYY).■";D,M,Y
200 E2=M:D2=D:G2=Y:GOSUB600
210 P2=ABS(Z2-T):K2=J+1
220 PRINT"■";
230 PRINTTAB(8)"■┌──────────────────────────┐"
240 PRINTTAB(8)"│■■COMPATABILITY ANALYSIS■■│"
250 PRINTTAB(8)"└──────────────────────────┘■"
260 PRINT TAB((40-LEN(W$))/2);W$
270 PRINT TAB((40-LEN(X$))/2);X$;"■"
280 PRINT
290 PRINT"■"W$":■";
300 J=K1:GOSUB790:PRINT",";
310 M=E1:GOSUB780
320 PRINTF1","G1
330 PRINT"■■"X$":■";
340 J=K2:GOSUB790:PRINT",";
350 M=E2:GOSUB780
360 PRINTD2","G2"■■"
370 Z=P2
380 P3=ABS(INT(((Z/23)-INT(Z/23))*23))
390 S3=ABS(INT(((Z/28)-INT(Z/28))*28))
400 C3=ABS(INT(((Z/33)-INT(Z/33))*33))
410 P5=ABS(100-((2*P3)*(100/23)))
420 S5=ABS(100-((2*S3)*(100/28)))
430 C5=ABS(100-((2*C3)*(100/33)))
440 PRINTTAB(9)"■┌──────────────────────┐"
450 PRINTTAB(10)"■PHYSICAL.....■";
460 PRINT USING FO$;P5
470 PRINTTAB(10)"■SENSITIVITY..■";
480 PRINT USING FO$;S5
490 PRINTTAB(10)"■COGNITIVE....■";
500 PRINT USING FO$;C5
510 PRINTTAB(9)"■└──────────────────────┘"
520 A5=(P5+S5+C5)/3
530 PRINTTAB(10)"■AVERAGE......■";
540 PRINT USING FO$;A5
550 PRINTTAB(9)"■┌──────────────────────┐"
560 PRINT"■■■■■"
570 PRINT"■■   PRESS ANY KEY FOR ANOTHER ANALYSIS  ■";
580 GETKEY A$
590 PRINT"■";:RUN
600 Y1=Y-1800
610 Q1=INT(Y1/4)
```

Commodore

Compatibility Analysis continued.

```
620 Q2=INT(Q1/25)
630 Q3=INT((Y1+200)/400)
640 K=0
650 IFQ1*4<>Y1THEN690
660 IFQ2*100<>Y1THEN690
670 IFQ3*400-200<>Y1THEN690
680 K=1
690 T=365*Y1+Q1-Q2+Q3-K
700 T=T+A(M)+D-1
710 IFM<3THEN730
720 T=T+K
730 IFINT(Y1/4)<>Y1/4THEN760
740 IFM>2THEN760
750 T=T-1
760 J=T-7*INT(T/7)
770 RETURN
780 PRINTM$(M);:RETURN
790 PRINTD$(J);:RETURN
800 FORJ=1TO12:READM$(J):NEXT
810 FORJ=1TO7:READD$(J):NEXT:RETURN
820 DATAJAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP
830 DATAOCT,NOV,DEC
840 DATAWEDNESDAY,THURSDAY,FRIDAY,SATURDAY
850 DATASUNDAY,MONDAY,TUESDAY
860 REM
870 REM
880 REM-----------------
890 REM  CONTROL CODES
900 REM IN THIS LISTING
910 REM-----------------
920 REM "□ = CLEAR
930 REM "■ = DOWN
940 REM "■ = BLU
950 REM "■ = PUR
960 REM "■ = ORNG
970 REM "■ = GRN
980 REM "■ = FLASH ON
990 REM "■ = FLASH OFF
1000 REM-----------------
```

NOTES:

Microbee

MOTORBIKE TRIALS

Motorbike Trials simulates a rider travelling a twisting 100 kilometre course.

The rider starts at the top of the screen facing down. The 'A' key moves him to the player's right and the '\' key moves him to the player's left.

Every few kilometres the rider moves toward the bottom of the screen and is followed by another rider (you always control the front one). If your rider reaches the bottom of the screen, you've completed the course — it's very hard!

Good luck ... and watch for slow-moving cars.

Ken Rowe
Dernancourt SA

```
00090 GOTO 700
00095 CLEAR:RESTORE
00100 P=63488+49*16
00110 FOR A=P TO P+16*8-1
00120 READ B:POKE A,B
00130 NEXT A
00140 DATA 36,255,255,255,60,36,189,189,126,126,60,60,189,255,189,24
00150 DATA 15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15
00160 DATA 240,240,240,240,240,240,240,240,240,240,240,240,240,240,240,240
00170 DATA 0,16,16,16,40,56,124,108,186,68,56,16,16,16,0,0
00180 DATA 0,0,1,2,4,30,46,73,50,28,32,32,64,128,0,0
00190 DATA 0,0,128,64,32,32,124,114,73,42,20,4,2,1,0,0
00200 DATA 8,137,73,34,20,93,46,86,28,99,22,236,26,41,68,32
00210 DATA 15,15,15,15,240,240,240,240,15,15,15,15,240,240,240,240
00220 CLS: R=1:N=33:E=5
00230 GOSUB 520
00240 IF RND*1<.2 THEN LET R=INT(RND*3)
00250 K=K-1+R:Z=0
00280 IF PEEK(258)=28 THEN LET N=N+1:Z=2 ELSE IF PEEK(258)=1 THEN LET N=N-1:Z=1
00290 PCG:POKE 61440+(Q*64)+N,180+Z:NORMAL
00300 IF K>20 THEN LET K=20 ELSE IF K<-29 THEN LET K=-29
00320 IF RND*2<.112 THEN 460
00330 PRINT TAB(30+K);:PCG:PRINT"22";:NORMAL:PRINT SPC(E);:PCG:PRINT"33":NORMAL
00340 IF PEEK(61440+(64*Q)+N)<180 AND PEEK(61440+(64*Q)+N)>175 THEN 370
00350 S1=S1+1:IF S1=100 THEN GOSUB 800
00360 GOTO 240
00370 POKE 61440+(64*Q)+N,183:FOR I=1 TO 22:IF RND*1<.55 THEN OUT 2,59:OUT 2,65:
NEXT I ELSE NEXT I
00380 PLAY 0,1;22,5;19,5;22,5;19,5;22,5;19,5
00390 F=Q*100+INT(S1):CURS1:PRINT"YOU COMPLETED ";INT(FLT(F)/1500*100);" Km OF A
 100 Km COURSE."\"AND PICKED UP ANOTHER ";Q;" RIDERS"\"1] ANOTHER GAME"\"2] END"
00400 A1$=KEY$:IF A1$="" THEN 400
00410 IF A1$="1" THEN 95 ELSE IF A1$="2" THEN PRINT\"BYE":END
00420 GOTO 400
00430 PRINT TAB(30+K);:PCG:PRINT"22";:FORI=1TOE:PRINT"8";:NEXTI:PRINT"33":NORMAL
00440 PLAY5,5;5,5;7,5;9,5;7,5;5,5;5,5:CURS1:PRINT"YOU HAVE COMPLETED THE 100 Km
COURSE WITH 14 RIDERS"\"THAT IS AN EXCELLENT ACHIEVEMENT!"\"1] ANOTHER GAME"\"2]
 END":GOTO400
00450 END
00460 IF RND*1<.35 THEN 490
00470 W=INT(RND*(FLT(E))+1):PRINTTAB(30+K);:PCG:PRINT"22";:NORMAL:PRINT SPC(W-1)
;:PCG:PRINT"1";:NORMAL:PRINTSPC(E-W);:PCG:PRINT"33":NORMAL
00480 GOTO340
00490 IF RND*1<.5 THEN LET E=E+1 ELSE LET E=E-1
00500 IF E<4 THEN LET E=4 ELSE IF E>8 THEN LET E=8
00510 GOTO 330
00520 NORMAL: FOR I=1 TO 15:IF I=2THEN570 ELSE PRINT TAB(30);:PCG:PRINT"22";:NOR
MAL:PRINT SPC(5);:PCG:PRINT"33":NORMAL:NEXT I
00530 POKE 61440+N,180
00535 CURS5,2:PRINT"<== 'A'";:CURS54,2:PRINT"'\' ==>";:CURS15,16
00540 FOR I=1 TO5:FOR D=1 TO 25:OUT 2,59:OUT 2,65:FOR G=1 TO D
00550 NEXT G:NEXTD:NEXTI
00560 RETURN
00570 PRINT TAB(30);:PCG:PRINT"228888833":NORMAL:NEXT I
00700 CLS:PLAY 16,2;16,2;13,4;16,4;18,2;18,2;16,4;18,4;18,4;20,2;18,2;16,2;16,4
00710 INPUT "   INSTRUCTIONS ? ";T1$
00720 IF T1$="YES"ORT1$="yes"ORT1$="y"ORT1$="Y"THEN730ELSE100
00730 CLS: PRINT"                              ** MOTORBIKE TRIALS **"\"
      **********"\
00740 PRINT"  You are set a difficult 100 km course."\"Along the way you are joi
ned by more riders."\"You will lead these riders through the winding corners"
00750 PRINT"And past slower cars."\\"The direction of the motorbike is to the bo
ttom of the screen"\"'A' moves the bike to your left & '\' to your right."
00760 PRINT\"The road changes width as well as direction."\\"If you can pick up
another 14 riders and reach the bottom of "\"the screen, you have completed the
course (very hard!!)"
00770 INPUT"                                          HIT <<RETURN>> TO START";T4$
00780 GOTO 100
00800 Q=Q+1:IF PEEK(258)=28THENLETN=N+1 ELSE IF PEEK(258)=1THENLETN=N-1
00810 S1=0 :IF Q=15 THEN 430
00820 RETURN
```

Microbee

CLOCK

Here is a program for all the egg burners among you. Just set the clock and alarm for perfect eggs, or use the stop watch for timing those three-hour chess games. The letter 'S' will stop and start the stop watch; press 'R' after the watch is stopped to return to the menu.

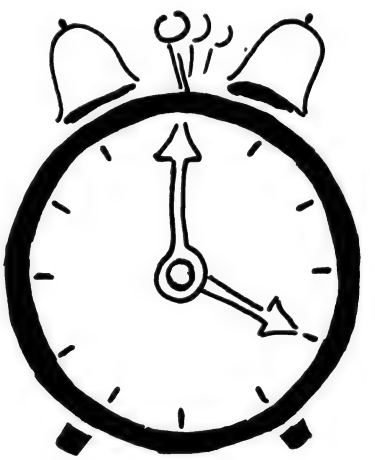
The program is quite easy to convert to other machines, given that 'Lores' and 'Hires' select graphics modes for the borders around the outside of the menu and the introduction. All 'Plots' can be left out, or your own computer's graphics commands may be substituted. CLS clears the screen. The Key$ is the same as Inkey$ on most other machines. The rest is fairly universal.

Jeremy Fenton
Lismore NSW

```
00100 REM ##################################################
00110 REM ##  CLOCK  BY J. FENTON                         ##
00120 REM ##  FOR PUBLIC USE                              ##
00130 REM ##  LAST UPDATE  2/3/85                         ##
00140 REM ##################################################
00150 LET X=0
00160 REM ################## INTRODUCTION ###################
00170 CLS
00180 HIRES
00190 PLOT 0,0 TO 0,255 TO 511,255 TO 511,0 TO 0,0
00200 PLOT 100,150 TO 50,150 TO 50,200 TO 100,200
00210 PLOT 120,200 TO 120,150 TO 170,150
00220 PLOT 190,200 TO 190,150 TO 240,150 TO 240,200 TO 190,200
00230 PLOT 310,200 TO 260,200 TO 260,150 TO 310,150
00240 PLOT 330,200 TO 330,150:PLOT 330,175 TO 390,200: PLOT 331,176 TO 390,150
00250 CURS 40,15:PRINT"PRESS ''RETURN''"
00260 A3$=KEY$:IF A3$="" THEN GOTO 260
00270 GOTO 720
00280 REM ################## SET CLOCK ######################
00290 CLS
00300 CLS:CURS 21,7:INPUT"ENTER TIME)-HOURS"H
00310 CURS 32,8:INPUT"-MINUTES"M
00320 CURS 32,9:INPUT"-SECONDS"S
00330 RETURN
00340 REM ################## CLOCK ######################
00350 CLS
00360 GOSUB 570
00370 GOSUB 620
00380 GOSUB 630
00390 GOSUB 640
00400 S=S+1
00410 FOR A0=1 TO 31 STEP.1007899
00420 NEXT A0
00430 GOSUB 640
00440 IF S<60 THEN 400
00450 S=0
00460 GOSUB 640
00470 M=M+1
00480 GOSUB 630
00490 IF M<60 THEN 490
00500 M=0
00510 GOSUB 630
00520 H=H+1
00530 GOSUB 620
00540 IF H<13 THEN 490
00550 H=1
00560 GOTO 370
00570 CURS 20,7:PRINT"HOURS  MINUTES  SECONDS"
```

Microbee

```
00580 CURS 30,15:PRINT"PRESS ''R'' FOR MENU"
00590 IF Q=H AND K=M AND V=S THEN PLAY 1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
00600 RETURN
00610 REM ################ ALARM ######################
00620 CURS 21,8:PRINT H" ":RETURN
00630 CURS 29,8:PRINT M" ":RETURN
00640 CURS 37,8:PRINT S" ":IF Q=H AND K=M AND V=S THEN PLAY 1;1;1;1;1;1;1;1;1;1;1;1;1;1
00650 A5$=KEY$:IF A5$="R" OR A5$="r" THEN GOTO 720
00660 RETURN
00670 REM ################ SET ALARM ######################
00680CLS: CURS21,8:INPUT"HOUR )";Q
00690 CURS 21,9:INPUT"MINUTES )";K
00700 CURS 21,10: INPUT"SECONDS )";V
00710 RETURN
00720 CLS
00730 REM ########## MENU ##############################
00740 LORES:PLOT 0,0 TO 0,47 TO 127,47 TO 127,0 TO 0,0
00750 CURS 30,5:PRINT"MENU"
00760 CURS 10,7: PRINT"SET THE CLOCK : S"
00770 CURS 10,9: PRINT"STOP WATCH    : W"
00780 CURS 10,11:PRINT"SET THE ALARM : A"
00790 CURS 10,13:PRINT"START CLOCK   : C"
00800 A5$=KEY$:IF A5$="" THEN GOTO 800
00810 IF A5$="S" OR A5$="s" THEN GOSUB 300
00820 IF A5$="a" OR A5$="A" THEN GOSUB 680
00830 IF A5$="C" OR A5$="c" THEN GOTO 350
00840 IF A5$="W" OR A5$="w" THEN GOTO 870
00850 GOTO 720
00860 REM ############## STOP WATCH ######################
00870 CLS
00880 CURS 20,6:PRINT" MINUTES   SECONDS ":GOTO 900
00890 FOR P=1 TO 800:NEXT P
00900 A5$=KEY$:IF A5$="S" OR A5$="s" THEN GOTO 970
00910 LET Z=Z+1
00920 CURS 34,0:PRINT Z
00930 CURS 25,8: PRINT X
00940 A5$=KEY$:IF A5$="S" OR A5$="s" THEN GOTO 970
00950 IF Z=60 THEN LET Z=0: X=X+1: GOTO 870
00960 GOTO 890
00970 A5$=KEY$:IF A5$="" THEN GOTO 970
00980 IF A5$="S" OR A5$="s" THEN GOTO 950
00990 IF A5$="R" OR A5$="r" THEN LET Z=0:LET X=0:GOTO 720
01000 GOTO 970
```



VZ200

NUMBER SEQUENCE

This program prints various sequences of numbers, each ending with a blank. You must enter the next number in the sequence — the computer indicates if your entry was correct.

A series of ten questions is asked, then your score is given.

Because the program is written in standard Microsoft BASIC, it should be easily transported to other computers. The random number statements in lines 120-140 may need modification, according to your particular version of BASIC.

Ian Thompson
Collaroy Plateau NSW

```
1 '*****************************
2 '*     NUMBER SEQUENCE       *
3 '* FOR THE UNEXPANDED VZ-200 *
4 '* IAN THOMPSON - COLLAROY   *
5 '*****************************
10 CLS:PRINT@104,"NUMBER SEQUENCE"
12 PRINT@325,"IAN THOMPSON,COLLAROY"
15 PRINT@485,"PRESS ANY KEY TO START"
20 IF INKEY$="" THEN 20
21 IF INKEY$="" THEN 20
25 CLS:PRINT"         NUMBER SEQUENCE":PRINT
30 PRINT"THIS PROGRAM WILL PRINT VARIOUS"
35 PRINT"SEQUENCES OF NUMBERS, EACH      "
40 PRINT"ENDING WITH A BLANK (----)."
45 PRINT"WHEN YOU SEE A '?', TYPE IN THE"
50 PRINT"NUMBER THAT YOU THINK THE "
55 PRINT"COMPUTER MIGHT HAVE PRINTED IN "
60 PRINT"PLACE OF THE BLANK."
70 PRINT
75 PRINT"*********************************"
80 LET R=0
90 LET W=0
100 FOR I=1 TO 10
110 PRINT"PROBLEM";I
120 LET A=INT(10*RND(0)+1)
130 LET B=INT(10*RND(0)+1)
140 LET G=RND(3)
150 IF A>B THEN 285
160 IF G=1 THEN 170
162 IF G=2 THEN 210
164 IF G=3 THEN 250
170 LET X=2*A+3*B
180 PRINT A;",";B;",";A+B;",";A+2*B;", ----";
190 INPUT Y
200 GOTO 410
210 LET X=A*A*B*B*B
220 PRINT A;",";B;",";A*B;",";B*A*B;", ----";
230 INPUT Y
240 GOTO 410
250 LET X=-B
260 PRINT A;",";B;",";B-A;",";-A;", ----";
270 INPUT Y
280 GOTO 410
285 IF G=1 THEN 300
290 IF G=2 THEN 340
300 LET X=A*5
310 PRINT A;",";2*A;",";3*A;",";4*A;", ----";
320 INPUT Y
330 GOTO 410
340 LET X=16*A
350 PRINT A;",";2*A;",";4*A;",";8*A;", ----";
360 INPUT Y
410 IF X=Y THEN 450
420 PRINT"NO; THE COMPUTER'S SEQUENCE HAS ";X;"."
430 LET W=W+1
440 GOTO 470
450 PRINT"THAT'S RIGHT!"
460 LET R=R+1
470 PRINT
480 NEXT I
485 SOUND 15,5
490 PRINT"================================="
500 PRINT"SCORE: ";R;" RIGHT,";W;" WRONG
505 PRINT:PRINT"================================="
510 PRINT" PRESS <SPACE> FOR ANOTHER SET   OF";
520 PRINT" QUESTIONS."
530 A$=INKEY$:IF A$ <> " "THEN 530
535 RUN
```

TRS-80

MAILIST

Mailist is a mailing list program for TRS-80 Models I, III and IV with 16 Kbytes of RAM. It stores up to 250 names and addresses on cassette tape for later retrieval and printing.

For those of you who have machines with 48 KBytes of RAM, a version for up to 1100 names and addresses with a built-in lowercase driver is available for $10, cassette supplied.

Both programs are menu-driven and a help sequence is included in the program (function 8). If you have any questions or wish to purchase the 48 Kbyte version, write to David Minehan, 64 Young Road, Lambton East 2299.

David Minehan
Lambton East, NSW

```
10 ' MAILIST.
20 ' PROGRAMMER: DAVID KEITH MINEHAN.
30 ' SYSTEM: TRS-80/SYSTEM-80.
40 ' RAM: 16K.
50 ' LANGUAGE: BASIC.
60 ' PROGRAM DESIGNATION: BUSINESS.
70 '
80 ' PROGRaM DESIGNED TO TAKE NAMES,
90 ' ADDRESSES, AND PHONE NUMBERS &
100 ' STORE THEM ON CASSETTE-TAPE
110 ' DATA FILE(S).
120 '
220 CLEAR5500:CLS:POKE16396,23
    :DIMN$(250),R$(250),A$(250),P$(250)
230
    PRINT TAB(30);"MAILIST":PRINT:PRINT: ' ***MAIN MENU*** '
240
    PRINTTAB(10)"CREATE FILE(S).....1"
250 PRINTTAB(10)"VIEW   FILE(S).....2"
260 PRINTTAB(10)"UPDATE FILE(S).....3"
270 PRINTTAB(10)"SAVE   FILE(S).....4"
280 PRINTTAB(10)"CALL   FILE(S).....5"
290 PRINTTAB(10)"VERIFY FILE(S).....6"
300 PRINTTAB(10)"PRINT  FILE(S).....7"
310 PRINTTAB(10)"HELP.....8"
320 PRINTTAB(10)"QUIT.....9"
330 PRINT:PRINTTAB(10)"WHICH FUNCTION DO YOU WISH TO EXECUTE";:INPUTQ
340 ONQGOTO410,510,610,710,810,910,1010,1110,9999
350 CLS:GOTO230:    ' ***MENU END*** '
410 CLS:INPUT"WHEN READY, HIT <ENTER> (TO CLOSE FILE TYPE 9999 FOR NAME)";X:CLS
420 FORI=1TO250:PRINT:PRINT"ENTER NAME (LAST FIRST, NO COMMAS PLEASE)"
430 PRINT"THEN HIT THE <ENTER> KEY";:INPUTN$(I)
440 IFN$(I)="9999"THENP1=I:GOTO490
445 PRINT"ENTER NAME OF REPRESENTITIVE (<ENTER> FOR NIL)"
446 INPUTR$(I)
450 INPUT"ENTER STREET NUMBER & NAME (NO COMMAS)";A$(I)
460 INPUT"ENTER SUBURB/CITY & POSTCODE";P$(I)
470 IFFRE(X)<250GOTO490:IFFRE(X)=250THENPRINT:PRINT"FILE FULL"
    :PRINT
    :INPUT"PRESS <ENTER> TO RETURN TO MENU";X
    :CLS
    :GOTO230
480 NEXT
490 PRINT"FILE CLOSED -- ":INPUT"TO SEE MENU, HIT <ENTER>";X:CLS:GOTO230
510 CLS:FORI=1TOP1:PRINTN$(I):PRINTR$(I):PRINTA$(I):PRINTP$(I):PRINT:NEXT
520 PRINT:INPUT"TO RETURN TO MENU, HIT <ENTER>";X:CLS:GOTO230
610 CLS:PRINT"ENTER THE NAME FOR THE LINE YOU WISH TO CHANGE (NO COMMAS)"
620 INPUTN$
630 FORI=1TOP1:IFN$=N$(I)GOTO670
640 NEXT
650 PRINT:PRINT"NAME NOT IN FILE -- ":GOTO690
660 PRINT
    :PRINT"ENTER THE CORRECTED INFO. : NAME, ADDRESS, PHONE --"
670 INPUTN$(I),R$(I),A$(I),P$(I)
680 PRINT:PRINT"THE FILE NOW READS: "
    :PRINTN$(I):PRINTR$(I):PRINTA$(I):PRINTP$(I):PRINT
690 INPUT"FOR ANOTHER CORRECTION, TYPE 1, OTHERWISE TYPE 0";X
    :IFX=1THEN610ELSECLS:GOTO230
710 CLS:
    INPUT"PREPARE CASSETTE + PLAYER, WHEN READY HIT <ENTER>";X
```

TRS-80

Mailist continued.

```
720 PRINT:PRINT"COPYING..."
730 PRINT #-1,P1
740 FORI=1TOP1:PRINT #-1,N$(I),R$(I),A$(I),P$(I):NEXT
750 PRINT:PRINT"COMPLETE -- NOTE TAPE LOCATION PLEASE -- "
    :PRINT
760 INPUT"TO RETURN TO MAIN MENU, HIT <ENTER>";X
    :CLS:GOTO230
810 CLS:
    INPUT"PREPARE CASSETTE + RECORDER, WHEN READY HIT <ENTER>";X     :PRINT
820 PRINT"INPUTING..."
830 INPUT#-1,P1
840 FORI=1TOP1:INPUT#-1,N$(I),R$(I),A$(I),P$(I):NEXT:PRINT
850 INPUT"INPUTING OF DATA COMPLETE. TO SEE MENU PRESS <ENTER>";     X:CLS:GOTO23
0
910 CLS:PRINT"FILE VERIFICATION:":PRINT:PRINT
920 INPUT"NAME FOR SEARCH";N$:PRINT
930 FORI=1TOP1:IFN$=N$(I)THENPRINT"FILE FOUND -- ":GOTO960
940 NEXT
950 PRINT"FILE NOT FOUND -- ":INPUT"PRESS <ENTER> TO CONTINUE";
     X:CLS:GOTO230
960 PRINT:PRINTN$(I):PRINTR$(I):PRINTA$(I):PRINTP$(I)
         :INPUT"DO YOU WANT THE NAME PRINTED (Y/N)";X$
         :PRINT
         :IFX$="Y"THENLPRINTN$(I):LPRINTCHR$(24):LPRINTR$(I)
         :LPRINTCHR$(24):LPRINTA$(I):LPRINTCHR$(24)
         :LPRINTP$(I)
970 INPUT"PRESS 1 TO CONTINUE, 0 TO RETURN TO MENU";X
     :IFX=1THEN900ELSECLS:GOTO230

1005 CLS
1010 CLS:PRINT"PRINTING...":PRINT
1020 FORI=1TOP1:LPRINTN$(I):LPRINTR$(I):LPRINTA$(I):LPRINTP$(I):LPRINTCHR$(194):
NEXT
1030 PRINT:PRINT"PRINTING COMPLETE.":PRINT
1050 INPUT"TO SEE MENU HIT <ENTER>";X
     :CLS:GOTO230
1110 CLS:PRINTTAB(30);"HELP":PRINT:PRINT
1120 PRINT"CREATE ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO CREATE THE FILES TO BE USED
IN STORING THE NAMES, ADDRESSES, AND PHONE NUMBERS."
1130 PRINT
1140 PRINT"VIEW ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO SEE THE FILES EITHER JUST
CREATED, OR JUST TRANSFERED INTO MEMORY FROM TAPE."
1150 PRINT:INPUT"TO CONTINUE, PRESS 1, ELSE PRESS 0 TO RETURN
TO MAIN MENU";X
          :IFX=0THENCLS:GOTO230
          :IFX=1THENGOTO1160
1160 CLS:PRINT"UPDATE ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO CHANGE THE FILES JUST CREA-
TED (FOR TYPO'S), OR FROM TAPE, TO MAKE CHANGES TO PHONE NUM-
BERS OR ADDRESSE(S) AS NEEDED."
1170 PRINT
1180 PRINT"SAVE ADDRESS FILE(S):

THIS PROCEDURE ALLOWS THE USER TO SAVE THE FILE(S) JUST CREATED
OR THE CHANGES JUST MADE."
```

TRS-80

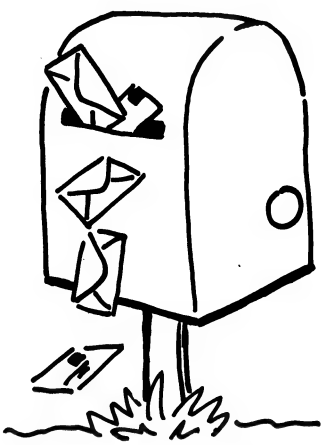
Mailist continued.

```
1190 PRINT
1200 PRINT:INPUT"TO CONTINUE, PRESS 1, ELSE PRESS 0 TO RETURN
TO MAIN MENU";X
          :IFX=0THENCLS:GOTO230
          :IFX=1THEN1210
1210 CLS:PRINT"CALL ADDRESS FILE(S):
THIS PROCEDURE ALLOWS THE USER TO CALL THE ADDRESS FILE(S) FROM
CASSETTE-TAPE DATA FILES."
1220 PRINT
1230 PRINT"VERIFY ADDRESS FILE(S):
THIS PROCEDURE ALLOWS THE USER TO VERIFY THE ADDRESS FILE(S).
THIS IS USED WHEN THE USER WISHES TO SEARCH FOR A PARTICULAR
NAME, ADDRESS AND PHONE NUMBER."
1240 PRINT:INPUT"TO CONTINUE, PRESS 1, ELSE PRESS 0 TO RETURN
TO MAIN MENU";X
          :IFX=0THENCLS:GOTO230
          :IFX=1THENGOTO1250
1250 CLS:PRINT"PRINT ADDRESS FILE(S):

THIS FUNCTION ALLOWS THE USER TO GET A HARD COPY OF THE FILES
EITHER ON PRINTER PAPER, OR STICKY LABELS FOR ENVELOPES."
1260 PRINT
1270 PRINT"HELP:

THIS FUNCTION."
1280 PRINT
1290 PRINT"QUIT:

END THE PROGRAM."
1300 PRINT
1310 INPUT"TO RETURN TO MENU, PRESS <ENTER>";X
     :CLS
     :GOTO230
9999 CLS:
      :POKE16396,201
      :END
```



TRS-80

MATH LAUNCH

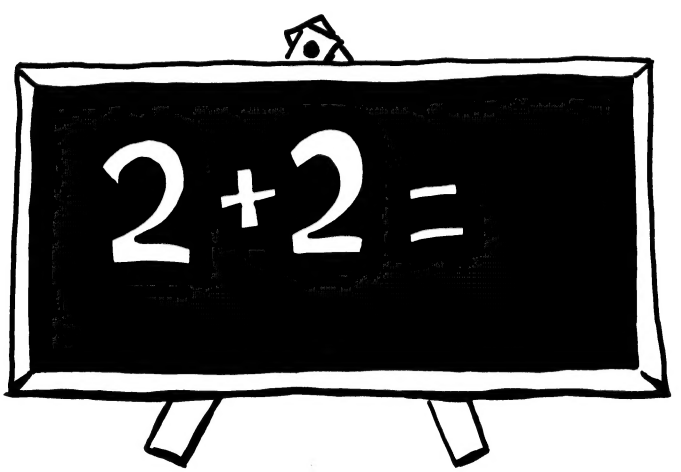
Math Launch is an educational game featuring high-res graphics, music and sound. The aim of the game is to build a rocket and launch it by answering multiplication questions. There is also a choice of three skill levels.

John Pospisil
Bulli NSW

```
85    CLS
7     CHAR63,0000003C003C
10    GOTO1000
506   NEXT
1000  GOSUB9000
1010  GOSUB9800
1015  CLS
1020  PRINT"WHICH SKILL LEVEL?"
1030  PRINT
1040  PRINT"    (A) VERY EASY"
1050  PRINT
1060  PRINT"    (B) INTERMEDIATE"
1070  PRINT
1080  PRINT"    (C) TOUGH"
1085  FORN=1TO10
1086  PRINT
1087  NEXT
1090  IFPEEK(29)=1THEN1100
1095  GOTO1090
1100  IFPEEK(25)=17THENA=7
1110  IFPEEK(25)=6THENA=15
1120  IFPEEK(18)=72THENA=40
1130  FORN=1TO20
1140  PRINT
1150  NEXT
1160  IFA=0THEN1020
1500  REM SREEN
1501  CHAR180,FFC3A59999A5C3FF
1502  FORN=22TO23
1503  FORM=14TO24
1504  PLOTN,M,180
1505  NEXT
1507  NEXT
1510  FORN=25TO27
1520  PLOTN,24,200
1530  NEXT
1540  FORN=18TO20
1545  PLOTN,24,252
1546  NEXT
1550  PLOT19,22,250
1555  PLOT19,23,251
1580  POKE219,208
1590  POKE218,228
1600  PRINT"LET'S START BUILDING !"
1610  FORN=1TO250
1620  NEXT
1630  POKE218,228
1640  PRINT"
1650  U=0
1800  REM QUESTION/ANSWER
1805  IFU=11THENGOSUB4300
1810  B=RND(A)
1820  C=RND(A+5)
1830  POKE219,208
1840  POKE218,230
1850  PRINTB;"  ";C;
1861  INPUTT$
1862  D=VAL(T$)
1870  IFD=C*BTHEN2000
1880  SOUND2;4,5;4,14;4
1890  SOUND2;3,5;3,14;3
1900  SOUND2;1,5;1,14;1
1910  SOUND2;4,5;4,14;4
1911  SOUND2;3,9;3,17;3
1912  SOUND2;1,7;1,16;1
1913  SOUND2;3,7;3,16;3
1914  SOUND2;1,5;1,14;1
1915  SOUND2;3,5;3,14;3
1916  SOUND2;1,4;1,12;1
1917  SOUND2;7,5;7,14;7
1920  FORN=13TO23
1930  FORM=25TO27
1950  PLOTM,N,32
1960  NEXT
1965  SOUNDN;0
1970  NEXT
1975  U=0
1980  POKE219,208
1985  POKE218,228
1986  PRINT"AH WELL,LETS START AGAIN"
1987  FORN=1TO300
1988  NEXT
1990  GOSUB5000
1995  GOTO1800
2000  REM
2005  IFU=11THEN4100
2010  GOSUB3000+U*100
2020  GOSUB5000
2030  U=U+1
2040  GOTO1800
2500  FORN=1TO256
2510  POKE4098,N
2520  FORM=1TO20
2530  NEXT
2540  NEXT
3000  PLOT25,23,96
3010  PLOT26,23,97
3020  PLOT27,23,98
3030  RETURN
3100  PLOT25,22,104
3110  PLOT26,22,105
3120  PLOT27,22,106
3130  RETURN
3200  PLOT25,21,107
3210  PLOT26,21,107
3220  PLOT27,21,107
3230  RETURN
3300  PLOT25,20,108
3310  PLOT26,20,109
3320  PLOT27,20,108
3330  RETURN
3400  PLOT25,19,108
3410  PLOT26,19,110
3420  PLOT27,19,108
3430  RETURN
3500  PLOT25,18,113
3510  PLOT26,18,112
3520  PLOT27,18,113
```

TRS-80

```
3530 RETURN
3600 PLOT25,17,113
3610 PLOT26,17,114
3620 PLOT27,17,113
3630 RETURN
3700 PLOT25,16,111
3710 PLOT27,16,111
3715 PLOT26,16,111
3720 RETURN
3800 PLOT25,15,120
3810 PLOT26,15,108
3820 PLOT27,15,121
3830 RETURN
3900 PLOT25,14,122
3910 PLOT26,14,124
3920 PLOT27,14,123
3930 RETURN
3999 END
4000 PLOT25,13,128
4010 PLOT26,13,129
4020 PLOT27,13,130
4030 RETURN
4100 FORN=18TO20
4110 PLOTN,24,32
4111 NEXT
4119 PLOT19,23,32
4120 PLOT19,22,32
4125 FORM=13TO24
4126 PLOT22,M,32
4127 PLOT23,M,32
4128 NEXTM
4130 GOSUB5000
4135 COLOR26,7,5
4140 CHAR200,FF7F7F7762
4145 POKE219,210
4150 POKE218,226
4155 POKE4098,0
4160 FORN=1TO24
4170 PRINT
4180 NEXT
4190 POKE219,209
4200 POKE218,68
4210 CHAR2,C0A0904F300C03
4220 CHAR3,000000F01010F0
4230 CHAR4,030509F20C30C0
4240 CHAR5,0000000F08080F
4250 PRINT"     WELL DONE"
4255 FORN=2TO10
4256 POKE4098,N
4260 FORM=1TO20
4270 NEXT
4280 NEXT
4290 SOUND0;0
4295 GOTO6000
4299 END
4300 FORN=1TO20
4305 POKE218,100
4310 PRINT"DATA FOR LAUNCH"
4320 POKE218,100
4330 PRINT"
4340 NEXT
4350 RETURN
5000 POKE219,208
5010 POKE218,228
5020 PRINT"
5030 RETURN
6000 SOUND26;3
6005 SOUND21;2
6010 SOUND21;1
6015 SOUND23;3
6020 SOUND21;3
6025 SOUND1;3
6030 SOUND9;3,19;3,25;3
6040 SOUND2;3,18;3,26;3
6050 FORN=1TO200
6060 NEXT
6070 PRINT"PRESS ANY KEY FOR NEW GAME"
6080 FORN=1TO10
6090 PRINT
6100 NEXT
6110 IFPEEK(29)=1THENRUN
6120 GOTO6110
8000 FORN=1TO255
8010 POKE4098,N
8020 FORM=1TO50
8030 NEXT
8040 NEXT
8999 END
9000 CHAR96,FF0F0F1F1F3F3F7F
9001 CHAR97,00FF5AA5FF
9002 CHAR250,010101030301E1E3E
9003 CHAR251,3E76F606060F0F0F
9004 CHAR1,0081422418244281
9005 CHAR252,FFFFFFFFFFFFFFFF
9010 CHAR97,FFFFFFFFFFFFFFFF
9020 CHAR98,FFF0F0F8F8FCFCFE
9030 CHAR104,FF0303030303
9040 CHAR105,FFC3C3C3C3C3
9050 CHAR106,FFC0C0C0C0C0
9060 CHAR107,000000FFFFFFFFFF
9070 CHAR108,00
9080 CHAR109,00003C3C18181818
9090 CHAR110,1800001C20180438
9100 CHAR112,2424000024242424
9110 CHAR113,00
9120 CHAR114,000000000018243C
9125 CHAR111,F0F0F0F00F0F0F0F
9130 CHAR120,0F0F1F1F3F3F7FFF
9140 CHAR121,F0F0F8F8FCFCFEFF
9150 CHAR122,0F0F0F0F0F0F0F0F
9160 CHAR123,F0F0F0F0F0F0F0F0
9170 CHAR124,FFC3C3C3FFFFFFFF
9180 CHAR128,0000000003030F0F
9195 CHAR129,003C7EFFFFFFFFFF
9200 CHAR130,00000000C0C0F0F0
9205 CHAR200,0000FFA55AFF
9210 RETURN
9800 COLOR0,13
9810 COLOR13,2,5
9820 COLOR14,2,16
9830 COLOR15,2,15
9835 COLOR16,16,5
9840 COLOR17,2,5
9845 COLOR26,2,5
9846 COLOR32,11,5
9850 FORN=6TO12
9860 COLORN,2,5
9870 NEXT
9877 COLOR1,2,5
9880 COLOR5,2,5
9885 COLOR23,2,5
9900 RETURN
9997 POKE4098,208
9998 POKE4098,0
9999 POKE4098,207
```



INDEX

Programs are listed according to page numbers under the computers or languages for which they are written; these are listed alphabetically.

. . . an innovative new series of specialist magazines for today's homemakers . . .

THE Lifestyle SERIES

Are you looking for inspiration and information on ways to improve your home — design a new kitchen or bathroom, plan your outdoor living area or decorate the children's room? The Lifestyle Series has all the answers. Forthcoming titles include:

- *Bathrooms*
- *Swimming Pools*
- *Outdoor Living*
- *Bedrooms and Children's Rooms*
- *Apartments*
- *Renovations and Restorations*
- *Look for these exciting titles — at your newsagent now*

LIFESTYLE, THE SERIES THAT GIVES YOU MORE — MORE IDEAS, MORE INSPIRATION, MORE INFORMATION